due to its own scoring in 43 per cent of the cases; its opponents, however, lost possession through scoring in 24 per cent of the cases.

With intelligence of this type before him, the coach who has to play team A is in a position to weigh his strategy. Let us suppose that he has several small, aggressive players adept at stealing the ball. He might, for example, adopt a strategy of interception and attempt to achieve a more favorable floor-play exchange ratio than team A's former opponents. More aggressive tactics would make his players liable to a greater risk of committing fouls. Because of his depth in aggressive reserves, he would not be worried by the risk of having his players foul out. This policy would, however, cause team A to use up more of its possessions in foul shots which have only half the pay-off of field goals. This would also tend to tip the score in his favor by reducing the amount of effort

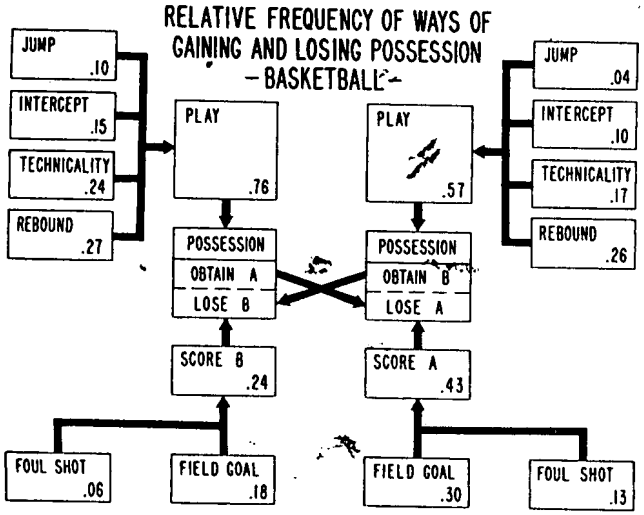


FIGURE 2

directed toward the higher pay-off of field goals. It seems evident from this exploratory study of basketball that it, too, presents many attractive features for the exercise of OR talents.

CONCLUSIONS

I am convinced that if coaches were to adopt the OR practices now employed in military operations, they would improve their strategy and tactics and the effectiveness of their teams. It would be necessary to begin with the pre-season training, which corresponds to the combat readiness phase of military operations, and keep records of performance as a basis for the development of measures of effectiveness. Scouts should keep similar records on the opponents. It would then be possible, with the appropriate models of the play and probability distributions, to work out the best strategy and tactics through the modern techniques of operational experimentation.

A COMMENT ON EDIE'S "TRAFFIC DELAYS AT TOLL BOOTHS"

GEORGE B. DANTZIG

The Rand Corporation, Santa Monica, California

(Received June 21, 1954)

E. W. PAXSON of Rand has suggested that linear programming methods could be used as an alternative procedure in the scheduling section of Leslie C. Edie's interesting paper on "Traffic Delays at Toll Booths."* The purpose of this note is to elaborate on this suggestion.

Edie, in earlier sections of his paper, develops a method for determining booth requirements; he then states the scheduling problem as follows:

"This process resulted in a schedule of booths throughout the day, from which could be determined the total number of booth-hours required for the day. One more step remained in the problem, that of determining how many toll collectors were required to keep the scheduled number of booths open, and still permit toll collectors' personal and meal reliefs to be given within certain restrictions. These restrictions were (a) working periods of not less than 1 nor more than 3 hr between reliefs or ends of the collector's tour, (b) meal reliefs in the middle 4 hr of an individual's tour, and (c) starting times not earlier than 6 A.M. and quitting times not later than 12:30 A.M."†

We propose first to discuss the scheduling under the simplified assumption that the assignments are made ignoring the need for reliefs. (Later on we will say a few words about reliefs and an important variation of the procedure.) The effective day for each toll collector is given as 6¼ hr. However, including his relief, it will be supposed that it becomes a 7-hr effective day which has a standard pattern of 3½ hr before the meal, ½ hr for the meal, and 3½ hr after the meal. If now we let x_j be the number of men who start their tour at time j, where $j=0, 1, 2, \cdots, n$ corresponds to the half-hour intervals throughout the day and if we let $a_{tj}=1$ if $t=j, \cdots, j+6, j+8, \cdots, j+14$ (where $j+7$ corresponding to the meal period is omitted) and $a_{tj}=0$ otherwise, then the assignments must be chosen such that

$$\sum_{j=0}^{n} a_{tj}x_j \geq b_t, \qquad (t=1, 2, \cdots, n+14) \quad (1)$$

where b_t are the required number of toll booths for period t. The optimum assignment problem may then be stated as finding a set of non-negative x_j satisfying (1) such that the total number of men assigned is a minimum, i.e.,

$$\sum_{j=0}^{n} x_j = \text{Min.} \qquad (x_j \geq 0) \quad (2)$$

Computationally, the writer has discovered (in another connection) that the solution to system (1) and (2) can often be done by hand in a short time (1 to 2 hours) because the $a_{tj}=0$ or 1. (A special adaptation of the dual form of the

* LESLIE C. EDIE, "Traffic Delays at Toll Booths," *J. Opns. Res. Soc. Am.* **2**, 107 (1954).

† *Ibid.*, p. 136.

simplex algorithm* is employed that does not require knowledge of the inverse of the basis. Since the special procedure is unpublished, I will be happy to supply readers with further details if they write to me.)

Reliefs, other than meals, may be treated by introducing more alternative patterns of work with time-period gaps introduced into the pre-meal and post-meal reliefs. (This may necessitate going to 15-minute instead of half-hour intervals.) The effect would be to multiply the number of unknowns by k^2 where k is the number of different times when a relief can take place in the pre- or post-meal part of the tour. Mathematically, reliefs cause an increase in the number of unknowns in system (1) and (2) while preserving its essential structure. This need not add greatly to the work. For example, one could optimize using only a subset of the patterns, e.g., using only a relief in the middle of the early and late part of the tours and then (dropping the dual approach) introduce the other variables and their associated work patterns as a refinement of the restricted solution.

Fractional values for the optimum x_j are possible, although in the few cases tested this did not occur. Roundings (some up, some down) will produce a solution with the value of (2) larger than the minimum for the number of men required. Their difference is likely to be small. Only if the difference is large should there be any concern about accepting the rounded solution as the true optimum.

By slightly reformulating the concept of work patterns, it is possible to reduce this problem to a variant of the standard transportation-type† linear programming problem. Here, special procedures exist that permit solving very large systems rapidly. It has the following advantages: (a) The solutions will be in integers. (b) The work pattern idea is more general, inasmuch as the worker is not forced to fit the work pattern exactly if he is not needed. To illustrate, consider the following array:

Pattern	Period t						Total
	(1)	(2)	(3)	(4)	(5)	(6)	
1.1	x_{11}	—	x_{13}	x_{14}	—	x_{16}	w_1
1.2	x_{21}	—	x_{23}	x_{24}	—	x_{26}	w_2
2.1	x_{31}	x_{32}	—	x_{34}	x_{35}	—	w_3
2.2	x_{41}	x_{42}	—	x_{44}	x_{45}	—	w_4
2.3	x_{51}	x_{52}	—	x_{54}	x_{55}	—	w_5
3.1	—	x_{62}	—	x_{64}	x_{65}	—	w_6
—	—	—	—	—	—	—	w_0
Total	b_1	b_2	b_3	b_4	b_5	b_6	N

The work pattern 1.1 states that a worker has reliefs in periods (2) and (5). In the other periods he must work only if needed at a toll booth, i.e., if $x_{1t}=1$ means he works in the period t, $x_{1t}=0$ means he does not. Each work pattern, e.g., (1.1 and 1.2) is repeated a number of times since several workers may be assigned the same pattern. The number of repetitions must be guessed in advance; the only requirement being that it should be in excess of any likely number of similar assignments found in an optimum solution. The equations requiring solution are

$$\sum_j x_{it}=w_i, \qquad \sum_i x_{it}=b_t, \qquad \sum_i w_i=N,$$

where x_{it} is omitted if (i, t) corresponds to a relief period, where w_i is the number of time periods in a work pattern actually needed, b_t is the total number of toll gates required in period t, N is the number of time periods available in a given day from the entire work force, and w_0 is the amount not used. The variables x_{ij}, w_i must be chosen such that

$$1\geq x_{ij}\geq 0, \qquad w_i\geq 0, \qquad w_0=\text{Max},$$

i.e., the variables are so chosen that the force will be as free as possible for other work. It will be noted that the condition of $x_{ij}\geq 1$ is an upper bound on a variable. Here again a variant of the simplex algorithm exists which takes advantage of this type of condition.*

* George B. Dantzig, Alex Orden, and Philip Wolfe, "The Generalized Simplex Method for Minimizing a Linear Form under Linear Inequality Restraints," Rand Corporation, RM-1264, 5 April 1954; George B. Dantzig, "The Dual Simplex Algorithm," Rand Corporation, RM-1270, 3 May 1954.

† George B. Dantzig, *Activity Analysis of Production and Allocation*, "Application of the Simplex Method to a Transportation Problem," Koopmans (Ed.), 1951.

* George B. Dantzig, "Variables with Upper Bounds in Linear Programming," Rand Corporation, RM-1271, 22 March 1954.

WAITING LINE SUBJECT TO PRIORITIES

JULIAN L. HOLLEY

Melpar, Inc., subsidiary of Westinghouse Air Brake Company, Alexandria, Virginia

(Received May 20, 1954)

IN HIS ARTICLE, "Priority Assignment In Waiting Line Problems,"* Alan Cobham obtained very interesting and valuable waiting-time formulas both for single- and multiple-channel systems. His reasoning is similar in the two cases, but it is given in detail for the single-channel case, and for definiteness the ensuing remarks will be directed exclusively to that case.

In view of the importance of Cobham's result, it seems desirable to amplify the discussion in two respects. In the first place, we shall sketch an alternative approach that avoids his infinite iterative procedure and that applies the relevant probability principles in what appears to be the simplest possible way. In the second place, we shall examine the implications of Cobham's formula for the case of a single priority level, i.e., for the case with which the literature is almost exclusively concerned.

To avoid needless duplication we shall use at will the definitions and notation of the Cobham article (*q.v.*), and in addition we shall introduce the quantities

$$\psi_k=\lambda_k/\mu_k, \qquad (1\leq k\leq p) \quad (1)$$

$$\sigma_k=\sum_{i=1}^{k}\psi_i, \qquad (0\leq k\leq p) \quad (2)$$

* Cobham, *J. Opns. Res. Soc. Am.* **2**, 70 (1954).

Annotated Editorial Policy and Instructions to Authors

The purpose of *Interfaces* is to increase communications about the practice of Operations Research and Management Science. Consequently, the most appropriate topics are the use or application of OR/MS in business, industry, or government. Comments on how *Interfaces* can better serve its readers and on the material published are actively solicited.

Papers should be submitted in duplicate, double spaced, on one side only, to the Editor-in-Chief or the appropriate column/associate editor. Please consult the first issue of each volume for complete Editorial Policy and Instructions to Authors.

Published in November, February, May, and August by The Institute of Management Sciences and the Operations Research Society of America at 146 Westminster Street, Providence, Rhode Island 02903. Second Class Postage paid at Providence, R.I. and at additional mailing offices.

ISSN 0092-2102

INTERFACES
Vol. 9, No. 3, May 1979

0092-2102/79/0903/0001$01.25

TELEPHONE SALES MANPOWER PLANNING AT QANTAS

Adel Gaballa and Wayne Pearce

Planning Department, Qantas Airways Limited, Sydney, Australia 2000

ABSTRACT. When the traditional procedure for planning annual manpower requirements for the telephone sales reservation offices of Qantas Airways was replaced by a model utilizing queueing and integer linear programming techniques, savings in excess of US$235,000 were realized in staff reductions over a two-year period, and investigation into applications in several other Qantas service areas continues.

Relationships of staff size to waiting time and service time are also evaluated.

Introduction

This paper illustrates briefly the steps of a new manpower plan for the Qantas network of sales reservation offices. The plan is carried out at the beginning of each fiscal year and updated halfway through the year. First the number of telephone calls are forecast for monthly, daily, and half-hourly intervals. Second, the monthly staffing strength which should achieve the company's service standards within the restriction of industry agreements is built, from half-hour intervals into weekly schedules for a given month. The first part relies heavily upon the company's history and surveys of load patterns for varying units of time. For the second part, Operations Research techniques in conjunction with a computer optimize the required staffing needs, incorporating nonoperational considerations to construct a yearly manpower requirement plan.

The basic manpower planning problem

Telephone sales reservation offices, in common with other labor-intensive businesses, cannot consistently overstaff to meet occasional overflows in work demands. That is, overstaffing would maximize personnel costs for idle or under-utilized operators as well as maximizing customer service. In the highly competitive harsh economic climate facing most airlines today, the difference between operating in the red or the black can be the efficiency with which manpower requirements are planned. Consequently, most offices understaff slightly, in general, to meet normal work loads. Hence when a crisis occurs, either the regular work force serves longer hours or temporary help is employed. On the other hand, if a telephone sales reservation office is consistently understaffed or ill-equipped, potential customers (individuals trying to make reservations) might get a busy signal or have to wait an excessive period of time before being served. This can ultimately result in lost sales; i.e., the potential customer reserves with a different airline that provides better customer service.

PLANNING — CORPORATE; PRODUCTION SCHEDULING — WORK STUDIES

The traditional method

Before the OR group became involved in manpower planning, the determination of manpower requirements was solely the responsibility of front-line managers. Of course their recommendations had to be submitted to higher management, but there were no techniques developed to aid in arriving at optimum solutions.

Overstaffing was usually not recognized, and there was no way to determine the extent of the problem if recognized. Understaffing was identified by a high lost-call rate and numerous complaints from travel agents and the general public. The usual approach at that point was to determine the general growth in work load, in terms of number of telephone calls, that had taken place since the last increase in staff numbers, then divide by the number of calls an operator could be expected to handle. This crucial ratio, having been calculated using a point of time when there was no problem, judging from a low lost-call rate and lack of complaints, usually failed to take into account the extent that the work load was *likely* to increase. Hence future needs tended to be underestimated.

Although some recognition was given to peaks and troughs of business activity during the year, there was little attention given to variables such as training, sick leave, and the introduction of new staff.

The alternative method

Our aim was an approach which would provide a manpower requirement plan to effectively meet operational and nonoperational needs (such as union agreements and social needs of employees) for an entire fiscal year. The goal was to arrive at effective manpower requirement levels yet remain flexible enough so local managers could influence the outcome and compensate for particular circumstances.

Data collection

The first step involved building up a reliable history on the pattern of the work load (number of calls) by month for the previous two years within each particular office. Then, for a three-month period immediately preceding the financial year under study, the average daily distribution of calls by half-hour periods during a day was measured. This was calculated separately for three time units; weekdays, Saturdays, and Sundays, since different working hours and shorter average length of calls characterize Saturday and Sunday. These daily distributions were found to exhibit substantial variations by time of day but followed consistent patterns over a period of time. Since no significant variation by week in a given month was observed, it was assumed that the work load for each week of a given month could be considered to be the same.

The models and their output

The Interactive Forecasting System model into which the two years of monthly records were fed allowed the introduction of value judgements on likely yearly growth rates and the effects of any possible changes in seasonal or monthly fluctuations. The output was a forecast of the likely call volume by month for the next year.

A multiserver queueing model utilized the output above, as well as the daily distributions by half hour, under the conditions:

(1) average service time of 3.5 minutes for weekday calls, 2.5 minutes for weekend calls;

(2) a company standard of service that not more than 10% of calls wait more than 20 seconds before service commences;

(3) an exponential distribution of call arrivals within each half-hour time unit;

(4) an assumption of equal efficiency among agents.

The queueing model then produced a schedule of staff requirements for each half hour of each day of the week for the average week of a given month.

The first integer linear programming model used (Appendix I) consolidated these half hour requirements into optimum daily shift patterns satisfying the following operational conditions:

(1) Shifts start only on the hour and half hour.

(2) Shifts start during the hours of 0700-0930, plus one shift which starts at 1500, hence there are seven possible shifts.

(3) Length of shifts starting between 0830-0930 is 8½ hours with a one hour lunch break; all other shifts are 8 hours long with a half hour lunch break.

(4) Lunch breaks are scheduled over a 2-hour interval for the 8-hour shift and over a 2½-hour interval for the 8½-hour shift, commencing on the hour and half hour.

(5) Tea breaks do not need to be scheduled.

Output consisted of the following information for each day of the week:

(1) number of staff per shift,

(2) start and finish time per shift,

(3) lunch schedule per shift,

(4) total staff needed for the day.

A second integer linear programming model (Appendix II) produced an optimal weekly roster with minimum work force for the average week of every given month, permitting two consecutive days off to each employee. (A manual method was also developed which produces an optimal schedule without resort to a computer.)

All of these models were developed in-house and validated before being applied in the operational phase. Checks since they were introduced confirm that they are accurate in meeting real life needs. Explicit details on the models or the output will be provided by the authors upon request.

Yearly manpower requirement plan

The nonoperational needs are then incorporated into the plan. From a two-year history of monthly manpower levels and sickness leave, the number of extra employees required per month to cover sickness leave is calculated. The annual leave requirement is allocated in an optimal manner with respect to trough and peak months of activity to minimize the additional staff needed. Training and recruiting, necessitated by natural attrition as well as business expansion, are scheduled throughout the year similarly.

Implications of varying customer waiting times and service times

Investigation into the implicit rules by which managers operated in times of crisis unearthed some significant relationships. It had been assumed heretofore that there were

occasions when it was worth staff savings to increase the average waiting time for customers. We found, on the contrary, that very small staff cutbacks resulted in considerable increase in average waiting times, due to the exponential shape of the curve (that is, tripling the waiting time of 20 seconds would reduce the required staff by only one). Hence increasing waiting time does not appear to be cost effective, particularly with its implication for public relations.

However, reducing the average service time of calls appears to be the most effective means of reducing staff size; cutting as little as half a minute from a 3.5-minute average call time could produce considerable savings. This area seems to be the most likely one for reducing manpower requirements for given call distribution and standard of service, through increasing staff efficiency and utilizing time-saving equipment.

Implementation and after

The original study concerned itself only with the major Qantas reservation sales offices in Sydney; in this, the OR group conducted the research and carried out the implementation, involving local staff only to the extent of utilizing their advice.

After the first study was completed, however, we pursued two new objectives: to ensure that local managers accepted responsibility for the final results, and to train local staff in the use of the techniques. Success in this "in-house" training varied from office to office; smaller offices tended to require less sophistication and the approach was consequently simplified. The organizational framework of the plan is based on a gradual disinvolvement of the OR group in favor of the office staff, who will prepare plans and submit them to the corporate personnel branch for approval. The OR role after the first year is an advisory one.

After the plan gained acceptance with the reservation sales office, demand developed for the same type of approach to be applied in other customer contact areas. Consequently, similar projects were undertaken for passenger sales offices and airport check-in facilities. These studies have further been used as partial input for planning future office space requirements.

Perhaps the most important impact of this work has been in acceptance of the general concept of manpower requirement planning, rather than simply communication of techniques. Managers have started to look for an optimum level of manpower based on a series of logical calculations, rather than a number with which to keep out of trouble for the next six months or so. The economic factors are becoming too complex for the "gut feeling" of managers to encompass.

Cost/Benefit

Research, computer time, and testing were estimated to cost US$35,000. The benefits reaped from one year's application in the Sydney office alone (1975-1976) were around US$173,000 (accomplished through resignations, transfers, and retirement).* This approach has remained in use in the Sydney office and has been adopted by other reservations sales offices as well, with further consequent savings.

* See letter confirming savings immediately following.

The future

We believe that this frame of thinking towards manpower planning of our airline's activities is going to prevail. Our optimistic view is derived, cynically enough, from the difficult economic environment within which airlines are operating; namely, half million dollar margins, which can be achieved as savings from this type of project. Whether from need or reason, Operations Research has a bright and sensitive role to play in the future of Qantas.

Acknowledgement

We are indebted to the then Sydney Reservations Manager, Mr. Malcolm Genge, for the success of this work. His firm support during the development stage led to gaining both the cooperation of reservations staff and approval of management for the project's implementation in Sydney and other major Qantas reservations offices.

Postlude

When this article was originally submitted for publication some time ago, we made a prediction that the methods contained in it for corporate manpower planning were in Qantas to stay; we are pleased to state that our prediction has proved accurate. Two cases where major manpower studies have been carried out according to our methods support this.

Case One. In June 1978 Qantas decided to study the economics of handling all telephone reservation sales in the state of New South Wales (which contains about 25% of Australia's population) from one central office in Sydney. This involved the possibility of closing three reservations offices in addition to the one in Sydney, and moving to a new and modern installation in North Sydney. Staff, equipment, and office space were calculated for the centralization according to the planning principles outlined in this article.

Case Two. A five-year plan for the gradual centralization of reservation sales covering the states of Western Australia, South Australia, Tasmania, and Victoria (65-70% of the population) to an office in Melbourne, Victoria was envisaged by management. Again, manpower, equipment, and office space were worked out as above, although the long-term planning was done with less detail.

Our implementation method involved the gradual handover of planning to reservation offices management, such that OR staff could remain in the background and provide advice only upon request. A colleague is currently preparing a full annual plan for our San Francisco office, and the centralized Sydney office is being helped to assess the impact of reduced air fares on manpower and equipment requirements.

The next logical step will be to make planning methodology available on an on-line interactive computer system.

QANTAS AIRWAYS LIMITED
(Incorporated in New South Wales)

70 Hunter Street Sydney N.S.W. Australia
Post Box 489 G.P.O. Sydney 2001

Telephone 2 0369 Area code 02
Telegraphic Qantas Sydney
Telex 20113

QANTAS

Ref

29 September, 1977

Prof. Gene Woolsey
Editor in Chief, Interface
Institute of Operations Research
Colorado School of Mines
Golden Colorado 80401, U.S.A.

Dear Professor Woolsey

The table below shows the actual savings resulting from the application of Manpower Planning Methods, stated in this paper, in the Sydney Telephone Reservations Office. The figures are for May-March 75/76 and the financial year 1976/77.

	Reservations Agents			Agents Group Leaders			Agents Supvrs			GRAND TTL SAV-INGS
	Man/ Mnth Redctn	Cost Per Mnth	Total Cost Redctn	Man/ Mnth Redctn	Cost Per Mnth	Ttl Cost Redctn	Man/ Mnth Redctn	Cost Per Mnth	Total Cost Red.	
		AUS$	AUS$		AUS$	AUS$		AUS$	AUS$	AUS$
1975/6	70	841	58,870	11	1,158	12,738	2	1,371	2,742	74,350
1976/7	93	937	87,141	12	1,262	15,144	17	1,530	26010	128295
TOTAL			146,011			27,882			28,752	202645

Regards,

R. K. Howe
Management Accountant - Corporate Branches
Australia's International Airline

Appendix I. Mathematical model for determination of optimal shift structure

A. Variables Definition:

S_j = number of staff on shift j, where $j = 1, 2, \ldots, 7$.

L_{ij} = number of staff from shift j taking their lunch break in period i, where $i = 1, 2, 3, 4$.

R_k = staff requirements for the time interval k (half-hour periods), where $k = 1, 2, \ldots, 30$.

B. Objective Function:

Minimize $\sum_{j=1}^{7} S_j$.

C. Constraints on the Shift Size:

$$S_j \geq 0, \quad j = 1, 2, \ldots, 7.$$

D. Constraints on the Relationships between Staff Requirements, Lunch Schedules, and Shift Sizes:

$$\begin{aligned}
S_1 &\geq R_1 \\
S_1 + S_2 &\geq R_2 \\
S_1 + S_2 + S_3 &\geq R_3 \\
S_1 + S_2 + S_3 + S_4 &\geq R_4 \\
S_1 + S_2 + S_3 + S_4 + S_5 &\geq R_5 \\
S_1 + S_2 + S_3 + S_4 + S_5 + S_6 &\geq R_6 \\
S_1 + S_2 + S_3 + S_4 + S_5 + S_6 - L_{11} &\geq R_7 \\
S_1 + S_2 + S_3 + S_4 + S_5 + S_6 - L_{21} - L_{12} &\geq R_8 \\
S_1 + S_2 + S_3 + S_4 + S_5 + S_6 - L_{31} - L_{22} - L_{13} &\geq R_9 \\
S_1 + S_2 + S_3 + S_4 + S_5 + S_6 - L_{41} - L_{32} - L_{23} - L_{14} &\geq R_{10} \\
S_1 + S_2 + S_3 + S_4 + S_5 + S_6 - L_{42} - L_{33} - L_{14} - L_{24} - L_{15} &\geq R_{11} \\
S_1 + S_2 + S_3 + S_4 + S_5 + S_6 - L_{43} - L_{24} - L_{34} - L_{15} - L_{25} - L_{16} &\geq R_{12} \\
S_1 + S_2 + S_3 + S_4 + S_5 + S_6 - L_{34} - L_{44} - L_{25} - L_{35} - L_{16} - L_{26} &\geq R_{13} \\
S_1 + S_2 + S_3 + S_4 + S_5 + S_6 - L_{44} - L_{35} - L_{45} - L_{26} - L_{36} &\geq R_{14} \\
S_1 + S_2 + S_3 + S_4 + S_5 + S_6 + S_7 - L_{45} - L_{36} - L_{46} &\geq R_{15} \\
S_1 + S_2 + S_3 + S_4 + S_5 + S_6 + S_7 - L_{46} &\geq R_{16} \\
S_2 + S_3 + S_4 + S_5 + S_6 + S_7 &\geq R_{17} \\
S_3 + S_4 + S_5 + S_6 + S_7 &\geq R_{18} \\
S_4 + S_5 + S_6 + S_7 &\geq R_{19}
\end{aligned}$$

$$
\begin{aligned}
S_4 + S_5 + S_6 + S_7 &\geq R_{20} \\
S_5 + S_6 + S_7 - L_{17} &\geq R_{21} \\
S_6 + S_7 - L_{27} &\geq R_{22} \\
S_7 - L_{37} &\geq R_{23} \\
S_7 - L_{47} &\geq R_{24} \\
S_7 &\geq R^*_{25},
\end{aligned}
$$

where R^*_{25} is the *minimum* staff requirement in the time intervals 25 to 30, i.e., from 1900 to 2200 hours. The average service time during this period of the day is usually smaller, hence to achieve maximum economy we took the minimum staff requirement during this period.

E. Lunch Times Constraints:

$$\sum_{i=1}^{4} L_{ij} - S_j = 0, \quad j = 1, 2, \ldots, 7.$$

These constraints ensure that every member of staff is scheduled for a lunch break.

F. Bounds on Variables:

$$
\begin{aligned}
S_j &= 0, 1, 2, \ldots \quad \text{for} \quad j = 1, 2, \ldots, 7 \\
L_{ij} &= 0, 1, 2, \ldots \qquad\quad i = 1, 2, 3, 4.
\end{aligned}
$$

(In effect, staff must be expressed in whole numbers.)

Appendix II. Formulation of the general case for scheduling variable staff requirements

A. Variables Definition:

r_j = minimum daily staff requirements for day j, where $j = 1, 2, \ldots, 7$.
W = total work force for the week ($W = 1/5 \sum_{j=1}^{7} r_j$), a 5-day work week.
$b_j = W - r_j$ = maximum number of staff allowed day j off.
$u = \frac{1}{2}(\sum_{j=1}^{7} y_j + t)$.
x_j = number of staff off on days j and $j + 1$ ($b_j = x_{j-1} + x_j$).
max r_j = maximum daily staff requirement during the week.
y_j = variations to the number of staff assigned day j off.
$t/5$ = number of extra staff needed due to rounding W, or because max r_j is greater than the average for the week ($1/5\Sigma_j r_j$).
W' = adjusted total weekly work force ($W' = W + t/5$).
$b'_j = b_j + y_j$ = adjusted maximum number of staff allowed day j off.
x'_j = adjusted number of staff allowed two consecutive days j, $j+1$ off.
$t = 5W' - \sum_{j=1}^{7} r_j$, the number of mandays needed per week.

B. Objective Function:

Minimize $\quad \frac{1}{2}\sum_{j=1}^{7} b_j + \sum_{j=1}^{7} y_j.$

C. Constraints:

$$x_{j-1} + x_j = b_j, \; 2 \leq j \leq 7 \qquad (1)$$

$$(x_7 + x_1 = b_1).$$

$$\sum_{i=1, i\in j}^{7} y_i \leq x_j + u - t/2, \quad j = 1, 2, \ldots, 7. \qquad (2)$$

$$\sum_{j=1}^{7} y_j = 2u - t. \qquad (3)$$

$$y_j \geq -b_j, \quad j = 1, 2, \ldots, 7. \qquad (4)$$

$$y_j \leq u, \quad j = 1, 2, \ldots, 7. \qquad (5)$$

D. Bounds:

$$u = \text{Maximum } \{0, \langle |(-\min x_j - t/2) \div 3| \rangle\},$$

where $\langle a \rangle$ denotes the smallest integer greater than or equal to the quantity a.

NOTICE FROM THE EDITOR

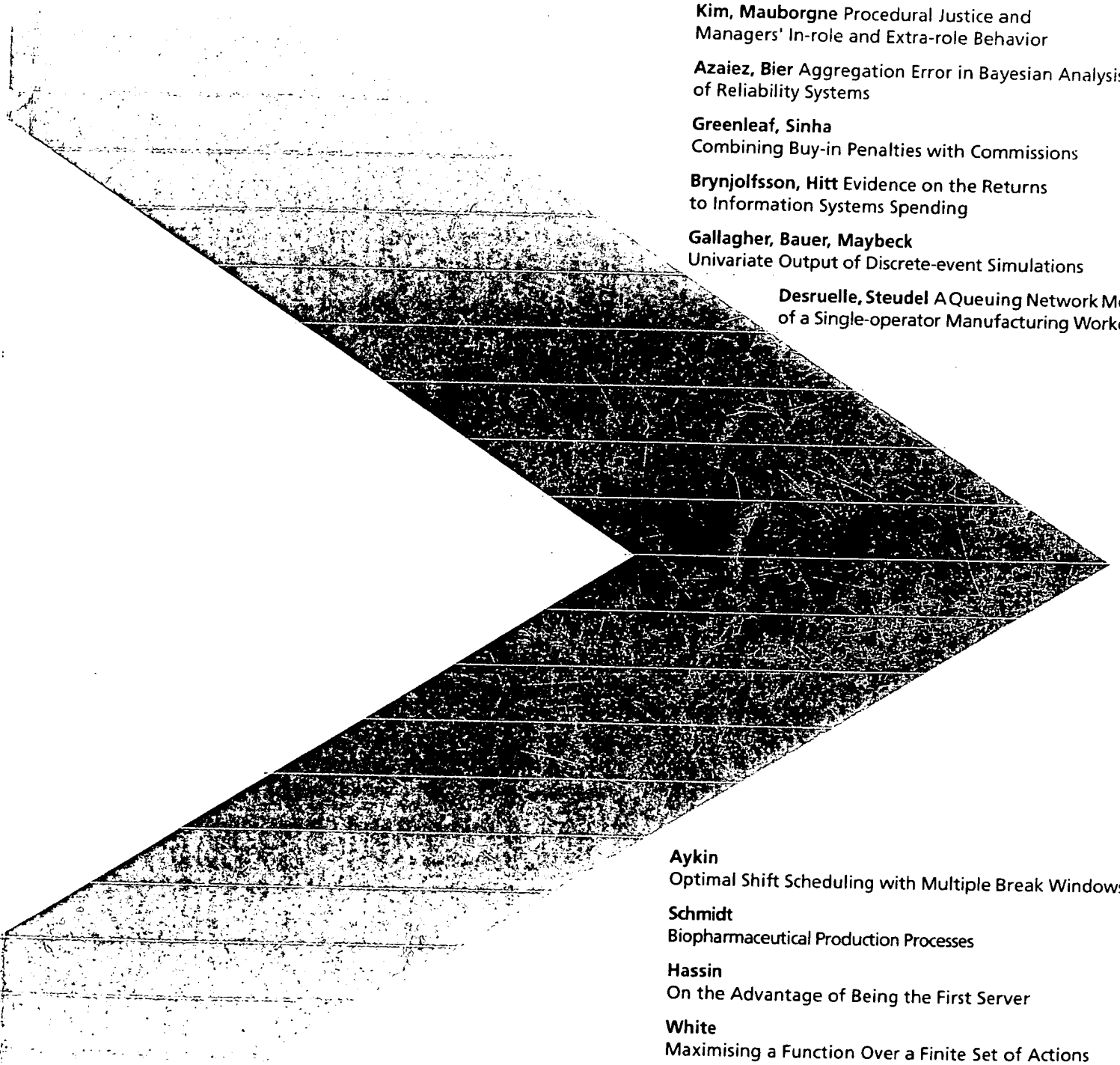
Both time and postage are wasted when mail is misdirected; please note the following instructions for *Interfaces* correspondence (addresses are on the Staff page overleaf of the Table of Contents):

All articles being submitted, revisions and inquiries thereon, letters to the editor, copyright agreement forms, camera-ready figures, corrected galleys, and requests to quote or reprint material should be sent to the Editor-in-Chief's office. Material being submitted for a column should be addressed to the appropriate Column or Associate Editor. Review copies of new books should be sent directly to the Book Review Editor. Queries on TIMS membership, back issues, and advertising should be addressed to the TIMS office, 146 Westminster Street, Providence, Rhode Island 02903. The address for subscription orders is on the application blank somewhere near the end of each issue.

Journal of the Institute for Operations Research and the Management Sciences

informs

MANAGEMENT SCIENCE



Volume 42, Number 4, April 1996

Optimal Shift Scheduling with Multiple Break Windows

Turgut Aykin
Graduate School of Management, RUTGERS, The State University of New Jersey, 180 University Avenue, Newark, New Jersey 07102

This paper presents a new integer programming model for optimal shift scheduling with multiple rest and lunch breaks, and break windows. A set-covering approach for this problem was originally developed by Dantzig (1954). Since then, a number of set-covering-based formulations have been proposed in the literature. These formulations require an integer variable for every shift type, shift start time, and rest/lunch break placement combination. Unfortunately, the number of integer variables required is rather large, making them impractical to solve for an optimal solution in most applications. We present a new approach in which a set of break variables is introduced for every shift-break type combination to determine the break placements. This approach leads to a significantly improved integer programming model requiring substantially smaller number of variables and computer memory. We tested the proposed approach with 40 test problems involving between 1,728 and 8,640 shift variations, and five demand patterns. Our results showed that the proposed formulation is very useful in solving large shift scheduling problems optimally.

(Manpower Scheduling; Service Operations Management; Integer Programming)

1. Introduction

The shift scheduling problem arises in a variety of service organizations such as telephone companies, airlines, hospitals, police departments, and banks and involves scheduling of manpower to meet demand that changes over the course of an operating day. Employees are assigned to various shifts specified by shift type, length (e.g., four-hour part-time, nine-hour full-time), shift start time, and the number and length of relief/lunch breaks. To provide flexibility in scheduling relief and lunch breaks, break windows (time intervals within which employees must start and complete their breaks) are often specified. Thus, the general shift scheduling problem involves determining the number of employees to be assigned to each shift and specifying the timing of their relief and lunch breaks. It is known that the flexibility provided in shift lengths, shift start times, and break placements lowers the total staffing cost and the number of employees needed (Showalter and Mabert 1988).

Since its introduction by Edie (1954), this problem and a number of related ones (e.g., the tour scheduling problem, the generalized employee scheduling problem) have attracted considerable attention in the literature. Both heuristic and integer programming based approaches have been proposed. The integer programming based approaches traditionally have been based on the following set-covering formulation originally suggested by Dantzig (1954).

$$\text{minimize} \quad \sum_{k \in K} c_k X_k, \tag{1}$$

$$\text{subject to} \quad \sum_{k \in K} a_{kt} X_k \geq b_t \quad \text{for all } t \in T,$$

$$X_k \geq 0 \quad \text{and integer}, \tag{2}$$

where K is the set of all shifts, T is the set of Zplanning periods that the shift schedule covers, b_t is the number

0025-1909/96/4204/0591$01.25

of employees needed in period t, c_k is the cost of assigning an employee to shift k, a_{kt} is equal to one if period t is a work period for shift k and zero otherwise, and X_k is an integer variable defined as the number of employees assigned to shift k, $k \in K$.

Given different shift types, lengths, shift start times, number and length of relief (rest) and lunch breaks and the time windows for these breaks, the set-covering formulation requires that all possible combinations of these features be included as different shifts in K. When relief and lunch breaks and break windows are included, the number of shifts in K increases rapidly. Consequently, standard integer programming tools fail to be useful. The difficulties caused by problem size are extensively discussed in the literature. These difficulties and the expected benefits from the added scheduling flexibility provided by break windows led researchers to study heuristic approaches to this problem. Segal (1974), for instance, considered a telephone operator scheduling problem in which, depending on the shift length, each shift may include up to two 15-minute relief and one 30-minute lunch break. Break windows are specified as 1.5 hours long so that each 15-minute relief break may start at six different times (assuming 15-minute planning periods). He proposed a network flow based heuristic for solving the shift assignment and the break placement problems separately in three phases. Keith (1979) considered the operator shift scheduling problem with lunch, early and late relief breaks at Illinois Bell Telephone Company. He proposed a slightly different set-covering model by ignoring break scheduling flexibility and assuming predetermined relief and lunch break times. The heuristic solution procedure proposed first solves the linear programming relaxation of the integer model and develops a feasible shift schedule using a rounding heuristic. Break windows are examined in the second part in an attempt to improve the shift schedule found. Keith (1979) reported that all Bell System Operating Companies were using a scheduling system based on either Keith (1979) or Segal's (1974) models. The shift scheduling problem with multiple relief and lunch breaks and break windows was also studied by Henderson and Berry (1976). They reported that, in their case, the number of different shift variations (in K) was 7120, and noted that there may be as many as 15,000 shift variations (Klasskin 1973) in more general cases. They proposed a number of heuristics to solve this problem in two phases. Other heuristic approaches for this problem are found in Buffa et al. (1976), Glover et al. (1984), Taylor and Huxley (1989), Thompson (1990), and Brusco and Jacobs (1993), among others. Besides these efforts, a number of researchers studied simpler cases involving either only one lunch break window (three start times for a lunch break, Bailey and Fields 1985) or fixed lunch break placement (one hour lunch break between the fourth and the fifth hours of work, Brusco and Jacobs 1993) while not including relief breaks.

While the solution aspect of the general shift scheduling problem has been attracting much interest, far less attention has been devoted to modelling options. Most studies used Dantzig's set-covering formulation or an extension thereof. Modelling a special case of the problem involving shifts with a single break has been considered by several researchers. Moondra (1976) considered a full-time shift with a lunch window allowing two break start times and part-time shifts working between four to eight hours without a lunch break. He represented the length of the part-time shifts implicitly and proposed a formulation assuming that 50% of the employees assigned to the full time shift will take their lunch break in the first break period (in the lunch window) and the remaining 50% in the second break period. More recently, Bechtold and Jacobs (1990) described an integer programming formulation using a set of integer variables to model break assignments. A variable of this type represents the total number of employees on all shifts starting their break in a particular planning period. In the problem they considered, employees are allowed a single (lunch) break. Their approach further assumes that the duration of the break is identical for all shifts and that every shift contains only a single break window consisting of contiguous planning periods. Moreover, the approach proposed in Bechtold and Jacobs (1990) is limited to cases involving less-than-24-hour operations and may fail to schedule breaks within their respective break windows in cases involving "extraordinary break-window overlaps" (described as a situation in which the break window for a shift is included entirely in the break window of another shift, and the larger break window starts at

least one planning period earlier and ends at least one planning period later than the smaller break window).

The number and the duration of breaks an employee takes are determined by many factors including legal restrictions, company policies, and union agreements. Typically, an employee working seven to nine consecutive hours a day receives one lunch break and two rest breaks, one before and one after the lunch. Shifts with shorter work span may be assigned fewer breaks. The length of a lunch break is usually a half hour to an hour and a rest break 15 to 30 minutes. Further, in some systems (e.g., telephone operator centers), employees may be assigned to split shifts consisting of two work periods separated by a break lasting two to four hours. A rest break is also provided in each work period. Thus, in many applications, the problem involves multiple breaks of unequal duration and multiple disjoint break windows. In this paper, we present a new integer programming formulation for this problem. To model the scheduling flexibility provided by the break windows, we introduce integer variables for the numbers of employees assigned to a shift and starting their breaks in different planning periods within the associated break windows. Thus, the break variables are associated with shifts and their values determine the break placements. Gaballa and Pearce (1979) considered a telephone sales system operating less than 24 hours assuming that the employees receive only one lunch break and *no* relief breaks. They proposed an integer programming formulation in which the scheduling flexibility provided by the lunch windows is modelled implicitly with additional integer variables. The main disadvantage of their model was that it required more integer variables and constraints than the equivalent set-covering formulation in the case they considered. Our formulation is also based on the implicit representation of the break placements. We consider the problem with multiple rest and lunch breaks and multiple break windows, and introduce different sets of break variables for different shift-break type combinations. More importantly, we show that, although not beneficial in the case considered by Gaballa and Pearce (1979), this approach leads to a significantly improved integer programming model requiring a substantially smaller number of variables than the equivalent set-covering model in the general shift scheduling problem involving multiple breaks and break windows. The number of nonzeros and the density of the A-matrix are also reduced significantly as compared to the set-covering formulation. In fact, the longer the break windows, the more efficient (compared to the set-covering approach) the new formulation becomes.

The proposed formulation is not restricted to a particular type of shift scheduling problem. Shifts may involve multiple rest and lunch breaks and multiple disjoint break windows. Breaks and break windows associated with different shifts may be different in number and length, and there may be break window overlaps. Like the set-covering formulation, the proposed approach is applicable to problems involving 24-hour continuous operations as well as less than 24-hour operations.

In the computational results section, we report our experience with test problems involving between 1,728 and 8,640 different shift variations using the integer programming option of LINDO, a general purpose ILP code. The largest problem we solved with the proposed approach using LINDO involved 32,928 shift variations (Aykin 1995). These represent, to the best of our knowledge, the largest shift scheduling problems solved to optimality in the literature. Although the size of the set-covering model suggests that the application of column-generation-based techniques may be helpful (for a column-generation-based heuristic for the tour scheduling problem, see Easton and Rossin 1991), this is yet to be studied. In contrast, the proposed approach provides a new model enabling the use of standard integer programming tools.

The remainder of the paper is organized as follows. In the next section, we discuss an example illustrating the proposed approach, define the notation and present the new formulation for the general shift scheduling problem. Computational results obtained with 40 test problems are presented in §3. In this section, we also discuss and present, for the first time, an optimal solution to a case reported in the literature by Segal (1974). Finally, §4 summarizes our results and discusses a number of research directions.

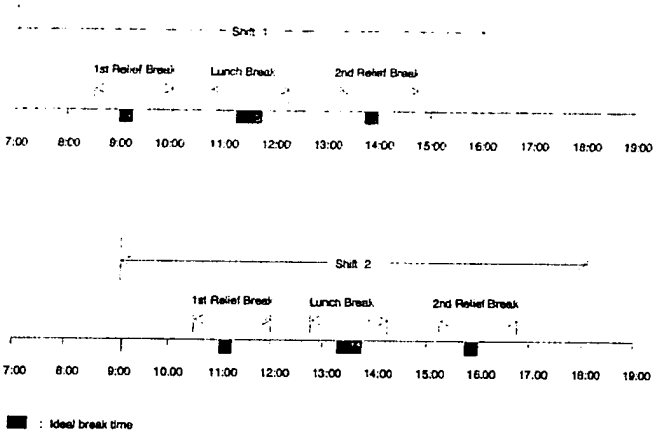
2. Problem Formulation

2.1. An Illustrative Example

To illustrate our approach, we consider a simple case involving only two shifts, $K = \{1, 2\}$, with a nine-hour shift span (defined as the duration of a shift including breaks). Employees are given one half-hour lunch break and two 15-minute relief breaks, leaving a work span of eight hours for each shift. Shift 1 starts at 7:00 and shift 2 starts at 9:00, and the system provides service from 7:00 to 18:00. Thus, there are 44 15-minute intervals in a work day, $t = 1, \ldots, 44$. Assume that the ideal start time for a lunch break is in the middle of the work span; that is, after four hours of work. The ideal relief break start times are in the middle of the four-hour work intervals; that is, two hours after the start of a shift and two hours after the completion of a lunch break starting at the ideal lunch time. Assume that all break windows are 1.5 hours long and starting half an hour before the ideal break start times. Then the time window for the first 15-minute relief break of shift 1 is from 8:30 to 10:00, for the lunch break from 10:45 to 12:15, and for the second 15-minute relief break from 13:15 to 14:45. Thus, the lunch break for an employee working this shift may be scheduled in five different ways: from 10:45 to 11:15, 11:00 to 11:30, 11:15 to 11:45, 11:30 to 12:00, and 11:45 to 12:15. And each 15-minute relief break may be scheduled in six different ways. The shifts, ideal break start times and the break windows are shown in Figure 1.

Let X_1 and X_2 be the numbers of employees assigned to shifts 1 and 2, respectively. Let U_{kt} be the number of employees working shift k and taking their first 15-minute relief break in period t. The employees assigned to shift 1 may take their first relief breaks in periods $t = 7, 8, 9, 10, 11, 12$ (that is 8:30, 8:45, 9:00, 9:15, 9:30, and 9:45), and the employees assigned to shift 2 in periods $t = 15, 16, 17, 18, 19, 20$. Similarly, define W_{kt} as the number of employees working shift k and starting their lunch break in period t. The employees assigned to shift 1 may start their lunch breaks in periods $t = 16, 17, 18, 19, 20$, and the employees assigned to shift 2 in periods $t = 24, 25, 26, 27, 28$. Define V_{kt} as the number of employees working shift k and taking their second relief break in period t. The employees working shift 1 may take their second relief breaks in periods $t = 26, 27, 28, 29, 30, 31$,

Figure 1 The Shifts, Ideal Break Times, and Break Windows in the Two-Shift Example



and the employees working shift 2 in periods $t = 34, 35, 36, 37, 38, 39$. The break variables U_{kt}, W_{kt}, and V_{kt} are defined only for break start times in the associated break windows. The objective function can be taken as to minimize total cost.

$$\text{minimize} \quad c_1X_1 + c_2X_2, \tag{3}$$

where c_1 and c_2 are the costs of assigning an employee to shifts 1 and 2, respectively.

If period t is in the shift span of k, then the number of employees assigned to shift k and working (i.e., not on break) in period t is given by

$$X'_{kt} = \begin{cases} X_k, & \text{if } t \text{ is not a break start time in any of the break windows,} \\ X_k - U_{kt}, & \text{if } t \text{ is a first relief break start time,} \\ X_k - W_{kt}, & \text{if } t \text{ is and } (t-1) \text{ is not a lunch break start time,} \\ X_k - W_{k(t-1)} - W_{kt}, & \text{if both } (t-1) \text{ and } t \text{ are lunch break start times,} \\ X_k - W_{k(t-1)}, & \text{if } (t-1) \text{ is and } t \text{ is not a lunch break start time,} \\ X_k - V_{kt}, & \text{if } t \text{ is a second relief break start time.} \end{cases} \tag{4}$$

Let b_t be the number of employees needed in period t and let a_{kt} be one if period t is in the shift span (a work or a break period) of shift k and zero otherwise. Then, the demand constraint for period t can be stated as

$$\sum_{s=k} a_{st} X_{kt} \geq b_t. \qquad (5)$$

The demand constraint for period 20, for instance, is

$$X_1 - W_{1,19} - W_{1,20} + X_2 - U_{2,20} \geq b_{20}, \qquad (6)$$

since period 20 is within two possible lunch breaks for shift 1 and within the first relief break window of shift 2.

In order to schedule two relief breaks and one lunch break for every employee, we include one constraint for every break window as:

$$X_k - \sum_{t \text{ in the 1st Relief Break Window of } k} U_{kt} = 0, \quad k \in K; \qquad (7)$$

$$X_k - \sum_{t \text{ in the Lunch Break Window of } k} W_{kt} = 0, \quad k \in K; \qquad (8)$$

$$X_k - \sum_{t \text{ in the 2nd Relief Break Window of } k} V_{kt} = 0, \quad k \in K. \qquad (9)$$

Although, with these break constraints, sufficient numbers of relief and lunch breaks are scheduled for all employees, their timing for an employee working a specific shift is not determined. The assignment of employees to specific combinations of scheduled breaks does not affect the total number of employees needed to meet the demand during a work day. The set-covering model, on the other hand, determines the number of employees assigned to every possible rest/lunch break placement combination explicitly and thus requires an excessive number of variables.

Employees working a shift can be assigned to specific relief and lunch break times specified with the break variables in a variety of ways. One approach may involve assigning employees to earliest first relief, lunch, and second relief break times available (the earliest break time rule). To illustrate, we solved the two-shift example described above using the part of the demand data given in Segal (1974) for the 44 15-minute planning periods between 7:00 and 18:00 (for an 11-hour work day covered by the two shifts). The optimal solution is shown in Table 1. In this solution, 21 employees are assigned to shift 1 and 87 employees to shift 2. The scheduled first relief, lunch and second relief break times (with the planning period index t within parentheses) together with the number of employees who will be on break at those times are also shown in Table 1. Using these break times and the earliest break time rule, the employees working shifts 1 and 2 can be assigned to specific break time combinations as shown in Table 2. Assigning employees to break combinations usually requires only a fraction of a CPU second and can even be done manually.

2.2. Integer Programming Model

To facilitate the presentation of the model, without losing generality we assume that every employee receives two relief breaks and one lunch break. Further, it is assumed that the duration of the relief breaks is one planning period and the duration of the lunch break is two planning periods. Let X_k be the number of employees assigned to shift $k \in K$, where K is the set of all shifts including all shift types/lengths, and allowed shift start times. Let U_{kt}, W_{kt} and V_{kt} be the numbers of employees assigned to shift k and starting their first relief break, lunch break, and second relief break in period t, respectively. Relief and lunch breaks must start and be completed within the specified time windows. Let $B1_k$, BL_k, and $B2_k$ be the sets of planning periods in which an employee working shift k may start his/her first relief break, lunch break, and second relief break, respectively.

The variables U_{kt1}, W_{kt2}, and V_{kt3} are defined for $k \in K$ and for $t1 \in B1_k$, $t2 \in BL_k$, and $t3 \in B2_k$, respectively. Breaks with duration longer than two planning periods (e.g., a two- to four-hour idle period allowed with split shifts) can be modelled similar to a lunch break using W_{kt}. In this case, however, every break variable W_{kt} is subtracted from the associated shift variable X_k in (5) for the duration of the break (e.g., for a four-hour break, subtracted from X_k in demand constraints for the 16 15-minute planning periods covered by the break). Shifts with fewer breaks (e.g., part-time shifts) can be modelled easily by excluding the variables and the constraint(s) associated with the unauthorized break(s). Shifts with more than three breaks can be included by introducing a new set of break variables and a new break constraint for every additional break. Let $T1_t$, TL_t, and $T2_t$ be the sets of shifts for which period t is a break start time within the time windows for the first relief break, lunch break, and second relief break, respectively. Let a_{st} be one if period t is in the shift span

Table 1 Optimal Break Start Times for the Second Shift Example

Shift (k)	Number of Employees Assigned	Break Times			Number of Employees on Break
		1st Relief Break U_{kt}	Lunch Break W_{kt}	2nd Relief Break V_{kt}	
1	21	9:15–9:30 (10)			21
			10:45–11:15 (16)		2
			11:45–12:15 (20)		19
				14:15–14:30 (30)	21
2	87	10:30–10:45 (15)			19
		10:45–11:00 (16)			11
		11:00–11:15 (17)			18
		11:15–11:30 (18)			20
		11:30–11:45 (19)			19
			12:45–13:15 (24)		23
			13:00–13:30 (25)		6
			13:15–13:45 (26)		33
			13:45–14:15 (28)		25
				15:15–15:30 (34)	19
				15:30–15:45 (35)	22
				15:45–16:00 (36)	22
				16:00–16:15 (37)	6
				16:15–16:30 (38)	6
				16:30–16:45 (39)	12

of shift k and zero otherwise. Then, the general shift scheduling problem can be formulated as follows:

$$\text{minimize} \quad \sum_{k \in K} c_k X_k, \tag{10}$$

subject to

$$\sum_{k \in K} a_{kt} X_k - \sum_{k \in T1_t} U_{kt} - \sum_{k \in TL_{(t-1)}} W_{k(t-1)} - \sum_{k \in TL_t} W_{kt} - \sum_{k \in T2_t} V_{kt} \geq b_t \quad \text{for all } t \in T, \tag{11}$$

$$X_k - \sum_{t \in B1_k} U_{kt} = 0 \quad \text{for all } k \in K, \tag{12}$$

$$X_k - \sum_{t \in BL_k} W_{kt} = 0 \quad \text{for all } k \in K, \tag{13}$$

$$X_k - \sum_{t \in B2_k} V_{kt} = 0 \quad \text{for all } k \in K, \tag{14}$$

$$X_k, U_{kt}, W_{kt}, V_{kt} \geq 0 \quad \text{and integer.}$$

The proposed formulation can be extended to model various restrictions on break placements, shift types, and objectives. The following constraint, for instance, may be added to limit the number of employees taking a (relief or lunch) break in period t.

$$\sum_{k \in T1_t} U_{kt} + \sum_{k \in TL_{(t-1)}} W_{k(t-1)} + \sum_{k \in TL_t} W_{kt} + \sum_{k \in T2_t} V_{kt} \leq h_t, \tag{15}$$

where h_t is the maximum number of employees who may be on break in period t. This constraint may be imposed if the number of employees taking a break in a period has to be limited for such reasons as space availability (e.g., cafeteria availability), fire regulations, or company policy.

The number of break variables U_{kt}, W_{kt}, and V_{kt} depends on the length of the break windows. The number of break constraints (12)–(14) is usually between 0 and 2 for a part-time shift, 3 for a full-time shift (for two relief and one lunch breaks), and 3 or 4 for a longer or a split shift. Thus the number of break constraints needed in the proposed formulation is determined by the number and types of shifts considered. Let the cardinality of set K be $|K| = n_K$. Also, let $|B1_k| = n_{1k}$, $|BL_k| = n_{Lk}$, $|B2_k| = n_{2k}$, and $|T| = n_T$. The number of vari-

Table 2 **Break Schedule for the Employees in the Second Shift Example**

Shift	Break Times: 1st Relief Break	Break Times: Lunch Break	Break Times: 2nd Relief Break	Number of Employees on Break
1	9:15–9:30	10:45–11:15	14:15–14:30	2
1	9:15–9:30	11:45–12:15	14:15–14:30	19
2	10:30–10:45	12:45–13:15	15:15–15:30	19
2	10:45–11:00	12:45–13:15	15:30–15:45	4
2	10:45–11:00	13:00–13:30	15:30–15:45	6
2	10:45–11:00	13:15–13:45	15:30–15:45	1
2	11:00–11:15	13:15–13:45	15:30–15:45	11
2	11:00–11:15	13:15–13:45	15:45–16:00	7
2	11:15–11:30	13:15–13:45	15:45–16:00	14
2	11:15–11:30	13:45–14:15	15:45–16:00	1
2	11:15–11:30	13:45–14:15	16:00–16:15	5
2	11:30–11:45	13:45–14:15	16:00–16:15	1
2	11:30–11:45	13:45–14:15	16:15–16:30	6
2	11:30–11:45	13:45–14:15	16:30–16:45	12

ables needed for a shift $k \in K$ in the proposed formulation is $(1 + n_{1k} + n_{Lk} + n_{2k})$ and in the set-covering formulation is $(n_{1k}n_{Lk}n_{2k})$ (assuming $n_{1k}, n_{Lk}, n_{2k} \neq 0$). The proposed formulation requires a total of $\{n_K + \Sigma_{k\in K} (n_{1k} + n_{Lk} + n_{2k})\}$ integer variables and $\{3n_K + n_T\}$ constraints (assuming three breaks for every shift in K), while the set-covering model requires $\{\Sigma_{k\in K} (n_{1k}n_{Lk}n_{2k})\}$ integer variables for all possible shift and break placement combinations and n_T constraints.

Note that different shift types do not have to be modelled following the same approach in the proposed formulation. If a shift is given only one break with a break window, enumeration of all shift variations would result in one less variable and no break constraint. Similarly, if the break times for a shift are predetermined then $n_{1k} = n_{Lk} = n_{2k} = 1$, and there is no need for break variables and constraints for this shift. The number of variables and constraints in the proposed model can be further reduced by using one of the Equations (12), (13), or (14) to replace X_k with a set of break variables. This approach, although it reduces the number of variables and constraints needed, increases the number of nonzeros and the density of the A-matrix. Our experience with this reduced formulation did not show any improvement in the computational effort needed to obtain an optimal schedule (Aykin 1995).

2.3. Upper Bounds

One can obtain upper bounds for decision variables X_t from the demand constraint (11). Since $X_t \geq U_{kt}$, $X_t \geq W_{kt}$ and $X_t \geq V_{kt}$, an upper bound on X_t also provides upper bounds on the break variables. If the employees working a shift, say $k \in K$, are not given any break but work SL_k periods, say periods $t = 1, \ldots, SL_k$, without loss of generality, then $X_k \leq b_{\max}$, where $b_{\max} = \max\{b_1, \ldots, b_{SLk}\}$. Thus, if $X_k = b_{\max}$, then employees working shift k will be sufficient to meet demand during periods $t = 1, \ldots, SL_k$. If the employees are given relief and lunch breaks, more employees may be needed during the break windows to meet demand and schedule these breaks. Suppose the employees assigned to shift k are given a one-period relief break with a break window $B1_k$. In a given period within the break window $t \in B1_k$, $(b_{\max} - b_t)$ employees are free to take a break. Thus, if $\Sigma_{t\in B1k} (b_{\max} - b_t) \geq b_{\max}$, then a relief break can be scheduled for every employee while meeting demand. Otherwise, let $\langle x \rangle$ be the smallest integer greater than or equal to x. Then,

$$\left\langle \left(b_{\max} - \sum_{t \in B1k} (b_{\max} - b_t) \right) \Big/ (n_{1k} - 1) \right\rangle$$

more employees will be needed during the break window in order for every employee to take his/her break and to meet demand. If this is the *only* break the employees assigned to shift k are given, then an upper bound for X_k is given by

$$UB1_k = \begin{cases} b_{\max} & \text{if } \sum_{t \in B1k} (b_{\max} - b_t) \geq b_{\max}, \\ b_{\max} + \left\{ \left\langle \left(b_{\max} - \sum_{t \in B1k} (b_{\max} - b_t) \right) \Big/ (n_{1k} - 1) \right\rangle \right\} & \text{otherwise.} \end{cases} \quad (16)$$

If the employees assigned to shift k are given a break lasting two or more periods (e.g., lunch break, idle period for a split shift), an upper bound on X_k can be obtained in a slightly different fashion. We illustrate this for a lunch break consisting of two consecutive periods with a break window BL_k. With $b_{\max}$ employees working

shift k, $(b_{\max} - b_t)$ employees are free to start their lunch break in period $t \in BL_k$. But since a break of this type lasts two periods, these employees can start their lunch breaks in period t if at least an equal number of employees are also free in period $(t + 1)$ and if $(t + 1) \in BL_k$. Hence, the maximum number of lunch breaks that can be scheduled within the lunch window with $b_{\max}$ employees is determined by examining the number of employees that are not needed in adjacent periods within that window. Let the maximum number of lunch breaks that can be scheduled with $b_{\max}$ employees be $L_{\max}$. If $L_{\max} \geq b_{\max}$, then a lunch break can be scheduled for every employee. Otherwise,

$$\langle (b_{\max} - L_{\max}) / ([(n_{Lk} + 1)/2] - 1) \rangle$$

more employees will be needed during the lunch window in order for every employee to take his/her lunch break. If this is the only break the employees assigned to this shift take, then an upper bound on X_k is given by

$$UBL_k = \begin{cases} b_{\max} & \text{if } L_{\max} \geq b_{\max}, \\ b_{\max} + \{\langle (b_{\max} - L_{\max}) / ([(n_{Lk} + 1)/2] - 1) \rangle\} & \\ & \text{otherwise,} \end{cases} \tag{17}$$

where $[x]$ is the largest integer not exceeding x.

If a break window permits only one break start time, then all employees should start their break at that time. Consequently, if the demand in that period is not zero, the above approach would not provide an upper bound.

When shift k includes multiple relief and lunch breaks, the highest of the bounds found for these breaks using (16), and (17) is taken as the upper bound for X_k. Upper bounds calculated this way were included in the proposed formulation as simple bounds. During our preliminary tests, these upper bounds have been found to be effective in reducing the computational effort needed to find an optimal schedule.

3. Computational Results

The computational experiments were conducted in two parts. In the first part, we studied the effects of break scheduling flexibility by analyzing a case reported in the literature by Segal (1974). Our results concerning this case are presented in §3.2. In the second part, we studied the effectiveness of the proposed model by solving 40 test problems using the integer programming option of LINDO. The results obtained with the test problems are presented in §3.3. The operating conditions assumed in our experiments are discussed in §3.1.

3.1. Scheduling Environment

The scheduling environment chosen for this study involves a continuous 24-hour work day (cyclical problem). Employee requirements are determined for 96 15-minute planning periods indexed successively $T = \{1, \ldots, 96\}$. Duration of a shift is taken as nine hours including breaks. Shifts are assumed to start on the hour or on the half hour, resulting in 48 shifts in a work day, $K = \{1, \ldots, 48\}$. Each employee is given one half-hour lunch break and two 15-minute relief breaks. Like in Segal (1974), we specify the break windows for each shift with reference to ideal break times as follows: the ideal start time for the first relief break is specified as two hours after the start of a shift, the ideal lunch break start time is located in the middle of the work day after four hours of work, and the ideal second relief break time is specified as after six hours of work. The break windows are formed by setting the earliest break start times half an hour before the ideal break times. The length of break windows is varied to create problems of varying size. The eight relief and lunch break window combinations considered are shown in Table 3. The relief break windows considered allow three to six break start times (45 minutes to 1.5 hour break windows) and the lunch break windows four to six lunch start times (1 hour and 15 minute to 1 hour and 45 minute lunch windows). In our computational experiments, we considered five different demand patterns with the eight break window combinations, resulting in 40 test problems. Three of the demand patterns were found in the literature and are either obtained directly from or resembling real patterns from service systems: (i) the trimodal demand profile for telephone operators described in Segal (1974) with a minimum quarter hour demand of 2 and a maximum of 89 operators, (ii) the bimodal demand profile with a minimum quarter hour demand of 2 and a maximum of 9 given in Henderson and Berry (1976) for the General Telephone and Electronics Company of California, and (iii) the unimodal

Table 3 **Set Covering versus Proposed Formulation: Problem Characteristics and Memory Requirements**

Break Window Combination	$n_{1k} = n_{2k}$	n_{Lk}	Number of Variables			Number of Constraints		Number of Nonzeros in A-Matrix			% Nonzeros in A-Matrix	
			Set Covering	Proposed Model	% Reduction	Set Covering	Proposed Model	Set Covering	Proposed Model	% Reduction	Set Covering	Proposed Model
1	3	4	1,728	528	69.5	96	240	55,296	3,024	94.5	33.3	2.4
2	3	5	2,160	576	73.3	96	240	69,120	3,168	95.4	33.3	2.3
3	4	4	3,072	624	79.7	96	240	98,304	3,216	96.7	33.3	2.1
4	4	5	3,840	672	82.5	96	240	122,880	3,360	97.3	33.3	2.1
5	5	4	4,800	720	85.0	96	240	153,600	3,408	97.8	33.3	2.0
6	5	5	6,000	768	87.2	96	240	192,000	3,552	98.2	33.3	1.9
7	5	6	7,200	816	88.7	96	240	230,400	3,696	98.4	33.3	1.9
8	6	5	8,640	864	90.0	96	240	276,480	3,744	98.6	33.3	1.8

demand profile with a minimum quarter hour demand of 2 and a maximum of 27 given in Buffa et al. (1976) for the General Telephone Company of California. In addition to these demand patterns, we considered two synthetically generated demand profiles: a bimodal and a trimodal sinusoidal demand pattern. Minimum and maximum quarter hour demand across these two demand patterns were 5 and 50, respectively.

3.2. Effects of Break Scheduling Flexibility

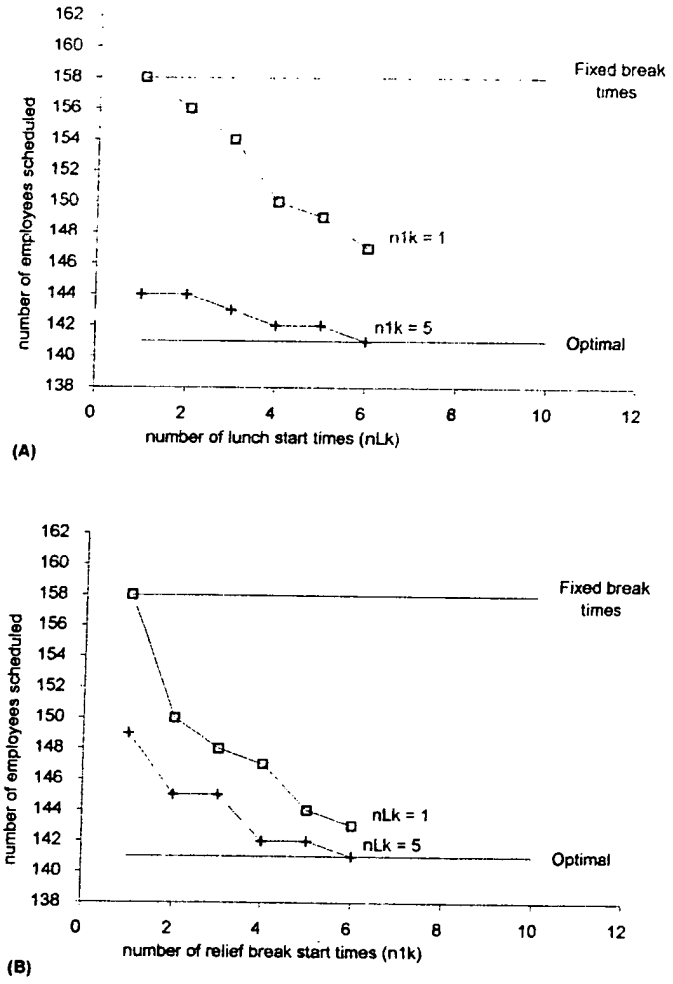
A number of researchers have demonstrated the effects of break scheduling flexibility in shift scheduling (see, for example, Bailey and Fields 1985). These studies, however, have remained limited to the cases involving shifts with a single break. To illustrate the effects of break scheduling flexibility in problems involving multiple breaks and break windows, we considered the demand profile given in Segal (1974) and varied the size of the break windows from one break start time (i.e., "fixed" break time) to six break start times. We first assumed the scheduling environment described in §3.1. and solved this problem repeatedly while changing the number of break start times. Figure 2A shows the optimal number of employees needed with different lunch window sizes when the relief break windows allow one or five relief break start times. As can be seen, compared to fixed relief break times, scheduling flexibility provided by five relief break start times lowers the number of employees needed significantly, between 6 to 14 employees or 4.1% to 8.9%, for all lunch window sizes. Figure 2B shows a similar pattern: for a given number of lunch break start times, the number of employees needed decreases as the numbers of relief break start times n_{1k} and n_{2k} increase. Moreover, Figures 2A and 2B show that when the employees are given multiple breaks, scheduling flexibility provided by the time windows specified for these breaks together affect the number of employees needed in the optimal schedule. In this case, when all three break times are fixed, 158 employees are needed. But, when the largest break windows are allowed (e.g., $n_{1k} = n_{2k} = 6$ and $n_{Lk} = 5$), the number of employees needed drops to 141, or 10.8% less.

Noting the importance of relief and lunch break placements in shift scheduling, Segal (1974) considered a set-covering formulation and developed a network flow based heuristic for this problem. He also discussed a case involving the demand pattern considered above and a number of four-, six-, seven-, eight- and nine-hour shifts. In this case, regular shifts lasting seven hours or longer included one half-hour lunch break and two 15-minute relief breaks while the shifts lasting 6 hours included only one half-hour lunch break and part-time shifts lasting 4 hours only one 15-minute relief break. All breaks had a 1.5-hour break window. The cost of assigning an employee to a six-, seven-, eight- or nine-hour shift was set to 2 and to a four-hour part-time shift to 1 (Segal 1993). He reported that the best solution found with his network flow based heuristic in this case had a total cost of 320 (only the total cost was reported for this solution). In this problem, the set-covering formulation requires 1478 integer variables and 96 constraints, while the proposed formulation requires only

Figure 2 Number of Employees versus Break Window Sizes

(A) Number of Lunch Start Times (n_{Lk}) with $n_{1k} = n_{2k} = 1, 5$

(B) Number of Relief Break Start Times (n_{1k}) with $n_{Lk} = 1, 5$



282 integer variables and 135 constraints. We solved this case with the proposed formulation and obtained an optimal schedule with a total cost of 298. Thus the cost of the schedule reported in Segal (1974) is 7.4% higher than optimal.

3.3. Experimental Results with the Test Problems

The number of variables, number of constraints, number of nonzero coefficients, and the density of the A-matrix for the set-covering formulation and the proposed model in the test problems considered are shown in Table 3. Compared to the set covering formulation, the proposed model requires substantially smaller, between 69.5% to 90% less, number of variables for all break window combinations. This reduction in the number of variables is achieved at the expense of three break constraints per shift (one per break). Net effect on the memory needed in terms of the number of nonzero coefficients in the A-matrix is even more remarkable—between 94.5% to 98.6% less than that for the equivalent set covering model. Further, note that the longer the break windows, the more efficient (compared to the set covering formulation) the proposed formulation becomes. In the largest case considered (break window combination 8 with $n_{1k} = n_{2k} = 6$ and $n_{Lk} = 5$), the set-covering model requires memory for 276,480 nonzero coefficients in the A-matrix, whereas the proposed model for only 3,744. Moreover, the density (percent of nonzeros) of the A-matrix for the proposed model is also substantially lower than the density of the A-matrix for the set-covering model: between 1.8% to 2.4% compared to 33.3% for the set covering formulation. Also note that as the break windows provide more flexibility with more break start times, the number of variables and the number of nonzeros in the proposed model increase at a much slower rate than for the set covering model. The density of the A-matrix, on the other hand, decreases. In summary, compared to the set-covering formulation, the proposed model for the general shift scheduling problem requires substantially smaller number of variables, significantly less memory for nonzero coefficients, and has a very low density.

The test problems were formulated using both the proposed formulation and the set-covering approach. All problems were solved using the integer programming option of LINDO with limits 20,003 columns, 14,900 integer variables, 6,503 rows, and 120,000 nonzero coefficients running in a batch mode on a VAX8550 computer.

Our test runs showed that nonzero break variables U_{kt}, W_{kt}, and V_{kt} tend to be integer when the associated shift variables X_k have integer values. Therefore, a branching strategy giving higher priority to shift variables X_k was implemented in LINDO. We also included the upper bounds calculated using (16) and (17) as simple bounds on the decision variables in LINDO. The CPU time limit was set to 750 seconds. No attempt was made to determine the branching requirements and total solution time if LINDO could not locate an integer

Table 4 Problem Generation, LP Solution and Integer Programming Solution Times (CPU Mins)

Break Window Combination	$n_{1k} = n_{2k}$	n_{Lk}	Average Problem Generation	Average LP Solution	Integer Programming Solution		
					Min	Average	Max
1	3	4	0.01	0.43	1.05	2.09	4.55
2	3	5	0.01	0.60	0.85	6.27	19.34
3	4	4	0.01	0.48	1.16	2.72	4.14
4	4	5	0.01	0.71	1.42	5.32	10.08
5	5	4	0.01	0.35	0.38	4.64	15.58
6	5	5	0.02	0.48	0.90	5.63	8.64
7	5	6	0.02	0.63	0.70	8.32	17.08
8	6	5	0.02	0.54	1.09	3.78	6.63

optimal solution to a problem within the CPU time limit. Instead, when this happened, a second attempt was made choosing variables for branching according to the total demand covered by the associated shifts. In this attempt, the shift variable X_k associated with a shift covering higher total demand was assigned a higher branching priority. In case of ties, the shift variable with the earliest start time was chosen for branching. If a second attempt was needed, the total time for the two attempts is reported. Thus, the CPU time limit for two attempts was 25 minutes.

The memory needed for the set covering formulation exceeded the memory limit of 120,000 nonzeros in all but ten cases involving the first two break window combinations. As seen in Table 3, the problems involving these break window combinations are the smallest considered in this study. Although the problems with break window combination 3 with $n_{1k} = n_{2k} = 4$ and $n_{Lk} = 4$ were inputted successfully into LINDO, the memory required consistently exceeded the memory limit while solving these problems and the runs had to be terminated. Out of the 10 test problems (with $n_{1k} = n_{2k} = 3$ and $n_{Lk} = 4$, and $n_{1k} = n_{2k} = 3$ and $n_{Lk} = 5$) that could be inputted, only three were solved within the CPU time limit in the first attempt. One more case was solved in the second attempt; the other six cases remain unsolved.

The average problem generation times including input to LINDO and average LP solution times in CPU minutes for the proposed formulation are shown in Table 4. The problem generation and input times were the same for the problems involving the same break window combination. The minimum, average, and maximum integer programming solution times in CPU minutes for the proposed formulation are also shown in Table 4. With the proposed model, out of 40 problems, an optimal schedule was obtained in 35 problems in the first attempt and in four more in the second attempt. In only one problem involving $n_{1k} = n_{2k} = 4$ and $n_{Lk} = 5$ an optimal schedule was not located within the allowed time. In the four cases solved with the set-covering formulation, solution times were between 4 and 14 times higher than the corresponding solution times with the proposed model. Finally, as seen in Table 4, with the proposed formulation, the integer programming solution time is not increasing as the number of break start times (break window sizes) increases. This is also consistent with our observation concerning the values of the break variables; the break variables U_{kt}, W_{kt}, and V_{kt}, whose numbers for a shift are determined by the numbers of break start times, tend to be integer when the associated shift variables X_k have integer values.

4. Extensions and Conclusions

In this paper, we presented a new formulation for the general shift scheduling problem with multiple breaks and break windows. Compared to the well-known set-covering formulation, the proposed model substantially reduces the number of variables needed at the expense of one additional constraint for each break–shift combination. Like the set-covering formulation, the proposed model is applicable to problems involving both 24-hour continuous operations

(cyclical problem) and less than 24-hour operations (acyclical problem), part-time, full-time, and split shifts involving multiple rest and lunch breaks, and multiple disjoint break windows.

We have discussed a number of practical considerations such as limits on the number of employees taking lunch and/or relief breaks in a planning period, and split shifts. An important extension of the proposed approach would involve its application to the tour scheduling problem (Aykin 1994). The tour scheduling problem involves both shift scheduling (including break scheduling for every shift) for the work days in a week and days-off scheduling. Another area in which more work needs to be done is the development and testing of specialized algorithms for the proposed formulation.[1]

[1] The author is grateful to Professor Awi Federgruen, the associate editor, and the reviewers for their constructive suggestions.

References

Aykin, T., "Reformulating the Tour Scheduling Problem to Improve Solvability," GSM Working Paper #94-25, Graduate School of Management, Rutgers, The State University of New Jersey, Newark, NJ, 1994.

——, "A Comparative Evaluation of Modelling Approaches to the Labor Shift Scheduling Problem," GSM Working Paper #95-01, Graduate School of Management, Rutgers, The State University of New Jersey, Newark, NJ, 1995.

Bailey, J. and J. Fields, "Personnel Scheduling With Flexshift Models," *J. Oper. Management*, 5, 3 (1985), 327–338.

Bechtold, S. E. and L. W. Jacobs, "Implicit Modeling of Flexible Break Assignments in Optimal Shift Scheduling," *Management Sci.*, 36, 11 (1990), 1339–1351.

Brusco, M. J. and L. W. Jacobs, "A Simulated Annealing Approach to the Cyclic Staff-Scheduling Problem," *Naval Res. Logistics Quarterly*, 40 (1993), 69–84.

Buffa, E. S., M. J. Cosgrove, and B. J. Luce, "An Integrated Work Shift Scheduling System," *Decision Sci.*, 7 (1976), 620–630.

Dantzig, G. B., "A Comment on Edie's 'Traffic Delays at Toll Booths'," *Oper. Res.*, 2, 3 (1954), 339–341.

Edie, L. C., "Traffic Delays at Toll Booths," *Oper. Res.*, 2, 2 (1954), 107–138.

Easton, F. F. and D. F. Rossin, "Sufficient Working Subsets for the Tour Scheduling Problem," *Management Sci.*, 37 (1991), 1441–1451.

Gaballa, A. and W. Pearce, "Telephone Sales Manpower Planning at Qantas," *Interfaces*, 9, 3 (1979), 1–9.

Glover, F., C. McMillan, and R. Grover, "A Heuristic Programming Approach to the Employee Scheduling Problem and Some Thoughts on 'Managerial Robots'," *J. Oper. Management*, 4 (1984), 113–128.

Henderson, W. B. and W. L. Berry, "Heuristic Methods for Telephone Operator Shift Scheduling: An Experimental Analysis," *Management Sci.*, 22, 12 (1976), 1372–1380.

Keith, E. G., "Operator Scheduling," *AIIE Trans.*, 11 (1979), 37–41.

Klasskin, P. M., "Operating to Schedule Operators," *Telephony*, July 30 (1973).

Moondra, S. L., "An L.P. Model for Work Force Scheduling for Banks," *J. Bank Res.*, Winter (1976), 299–301.

Nanda, R. and J. Browne, *Introduction to Employee Scheduling*, Van Nostrand Reinhold, New York, NY, 1992.

Segal, M., "The Operator-Scheduling Problem: A Network Flow Approach," *Oper. Res.*, 22 (1974), 808–823.

——, Personal Communication, June, 1993.

Showalter, M. J. and V. A. Mabert, "An Evaluation of a Full-/Part-Time Tour Scheduling Methodology," *International J. Oper. and Production Management*, 8, 7 (1988), 54–71.

Taylor, P. E. and J. Huxley, "A Break from Tradition for the San Francisco Police: Patrol Officer Scheduling Using an Optimization based Decision Support System," *Interfaces*, 19, 1 (1989), 4–24.

Thompson, G. M., "Shift Scheduling in Services When Employees Have Limited Availability: An L.P. Approach," *J. Oper. Management*, 9, 3 (1990), 352–370.

Accepted by Awi Federgruen; received May 3, 1994. This paper has been with the author 1½ months for 2 revisions.

ELSEVIER

European Journal of Operational Research 125 (2000) 381–397

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/dsw

Theory and Methodology

A comparative evaluation of modeling approaches to the labor shift scheduling problem

Turgut Aykin *

Engineering Research Center, Bell Laboratories, Princeton, NJ 08542, USA

Received 1 November 1996; accepted 1 August 1998

Abstract

The labor shift scheduling problem has traditionally been formulated using the set covering approach proposed by Dantzig, G.B. [1954. Operations Research 2 (3), 339–341]. The size of the resulting integer model with this approach, however, has been found to be very large to solve optimally in most practical applications. Recently, Bechtold, S.E., Jacobs, L.W. [1990. Management Science 36 (11), 1339–1351] proposed a new integer programming formulation requiring significantly smaller number of variables and nonzeros in the A-matrix than the equivalent set covering formulation. However, due to its assumptions, this approach has remained limited to the special cases of the shift scheduling problem. In Aykin, T. [1996. Management Science 42, 591–603], we presented another implicit modeling approach that is applicable to the general problem without any of the limitations of Bechtold and Jacobs (loc. cit.). This approach also requires substantially smaller number of variables and nonzeros in the A-matrix than the equivalent set covering formulation. In this paper, we relax the assumptions of Bechtold and Jacobs (loc. cit.) and present a generalized version of it. The extended formulation of Bechtold and Jacobs, although requiring smaller number of variables, has more constraints, more nonzeros in the A-matrix, and significantly higher density than the formulation of Aykin (loc. cit.). We compare these modeling approaches through solving (optimally) 220 problems involving as many as 32 928 shift variations. Our computational results show that the time needed to locate an optimal shift schedule and the percentage of the problems solved optimally with Aykin's formulation are substantially better than those with the formulation of Bechtold and Jacobs.

Keywords: Labor shift scheduling; Manpower management; Service operations management; Implicit modeling; Integer programming

1. Introduction

One of the critical tasks the manager of a service system faces daily is the effective scheduling of its manpower. In most service systems, demand for services changes during the course of an operating day. In order to develop a shift schedule, demand

* Correspondence address: 23 John st., Red Bank, NJ 07701. USA.

E-mail address: taykin@lucent.com (T. Aykin).

0377-2217/00/$ - see front matter
PII: S0377-2217(99)00413-0

needs to be forecasted and converted (usually with the help of queuing models) into labor requirements needed to maintain a desired service level during the course of an operating day. Employees are then assigned to various shifts specified by their start times, lengths, number and timing of relief and lunch breaks to meet these requirements. While understaffing will lower total labor cost, it will result in deterioration of the service quality and longer waiting times, and consequently, higher total cost. Overstaffing, on the other hand, will improve the service quality but will result in underutilization and excessive labor costs. Therefore, it is important for a service organization to schedule its manpower in an efficient manner to minimize labor costs while providing the desired service level.

The shift scheduling problem involves determining the number of employees to be assigned to various shifts and the timing of their breaks within the limits allowed by legal, union, and company requirements. To improve manpower utilization and lower labor costs, flexibility in shift types, start times, length, number and duration of breaks is provided. Flexibility in scheduling relief and lunch breaks for a given shift is provided with break windows specifying the time intervals within which employees assigned to that shift must start and complete their breaks. Since its introduction by Edie (1954), this problem has attracted considerable attention in the literature. Both heuristic and integer programming based approaches have been proposed. The integer programming based approaches have traditionally been based on the set covering formulation proposed by Dantzig (1954). For nearly four decades, this was the only modeling approach available for the general problem. Although the set covering formulation has been studied extensively, it is not a practical approach to solve this problem optimally. The set covering approach treats every possible shift type, shift start time, and break placement combination as a separate shift and requires a separate integer variable for it. As a result, the size of the integer model increases very rapidly with the number of shift types, shift start times, breaks, and break window sizes, making it impractical to analyze with the available integer programming tools.

Recently, two new modeling approaches have been developed by Bechtold and Jacobs (1990), and Aykin (1996). Both studies model break placements implicitly to reduce the size of the integer model. The objective of this study is to comparatively evaluate these two modeling approaches to the general shift scheduling problem in a variety of problem situations. For this purpose, we first relax a number of assumptions made in Bechtold and Jacobs (1990) and present a generalized version of their formulation that was used in the computational experiments reported. The modeling approaches are compared using a set of 220 test problems involving as many as 32 928 shift variations. The problem set considered in this study include, to the best of our knowledge, the largest shift scheduling problems solved optimally in the literature. Our results show that the formulation given in Aykin (1996) requires more variables than the formulation of Bechtold and Jacobs but substantially less than the set covering model. The number of constraints, number of nonzeros and the density of the A-matrix with the approach given in Aykin (1996) are, however, substantially smaller than those for the formulation given in Bechtold and Jacobs, and the set covering model. More importantly, our results show that the time needed to locate an optimal shift schedule and the percentage of the problems solved optimally with Aykin's formulation are substantially better than those with the formulation of Bechtold and Jacobs; while no optimal solution could be found in 32 (14.5%) problems with the formulation of Bechtold and Jacobs, only 2 problems remained unsolved with the formulation of Aykin. The average IP solution time for the problems solved by both modeling approaches was 331.36 seconds with the formulation of Bechtold and Jacobs, and 113.04 seconds (about 66% less) with the formulation of Aykin. The fact that these results were obtained with a problem set including the largest shift scheduling problems solved optimally in the literature using an off-the-shelf integer programming software (LINDO) is very encouraging in terms of the practical applications of implicit formulations in service systems.

The remainder of the paper is organized as follows. In Section 2, we introduce the set covering

formulation, extend the formulation given in Bechtold and Jacobs into the scheduling environments involving 24-hour continuous operations, multiple relief/lunch breaks, and multiple disjoint break windows, and present the formulation given in Aykin (1996). Computational results obtained with 220 test problems are presented in Section 3. Finally, in Section 4, we summarize our results.

2. Modeling approaches to the shift scheduling problem

This section introduces the notation used and presents the three modelings available for the general shift scheduling problem.

2.1. Set covering model of Dantzig (1954)

The following set covering formulation was introduced by Dantzig (1954):

$$\text{Minimize} \quad \sum_{k\in K} c_k X_k \tag{1}$$

subject to

$$\sum_{k\in K} a_{kt} X_k \geqslant b_t \quad \text{for all } t \in T,$$

$$X_k \geqslant 0 \text{ and integer}, \quad k \in K, \tag{2}$$

where X_k is an integer variable defined as the number of employees assigned to shift k, T is the set of planning periods covered by the shift schedule, K is the set of all shifts, b_t is the number of employees needed in period t to achieve the desired service level, c_k is the cost of assigning an employee to shift k, and a_{kt} is equal to one if period t is a work period for shift k and zero otherwise.

With this formulation, all shift type, shift start time, and break placement combinations are enumerated and included, as separate shifts, in set K.

2.2. Implicit modeling approach of Bechtold and Jacobs (1990)

Recognizing the difficulties caused by the enumeration of all shift variations in the set covering model and making use of the fact that not all information concerning the break placements included in that model is necessary to develop an optimal shift schedule, Bechtold and Jacobs (1990) developed an implicit formulation of the problem that is substantially smaller than the equivalent set covering model. Although the implicit modeling idea itself is not new in the manpower scheduling literature (see, for example, Moondra, 1976; Gaballa and Pearce, 1979), their formulation uses a novel approach for modeling break placements implicitly. Bechtold and Jacobs (1990) compared their formulation with the set covering model experimentally and reported superior results with their approach.

Unfortunately, due to its assumptions, this formulation is limited to certain instances of the problem and cannot be applied to the general case in its present form. The following assumptions are listed in Bechtold and Jacobs (1990, p. 1341): (1) the system operates less than 24 hours daily, (2) planning periods are equal in length, (3) each shift is given a single break, (4) the break duration is identical for all shifts, (5) the break duration is one or more periods, (6) each shift has a single break window associated with its workspan, (7) breaks should start and end during the shifts, (8) no extraordinary break window overlap exists (this condition exists when the break window of one shift is a strict subset of the break window of another, and has different starting and ending periods than the larger window), and (9) no understaffing is allowed. Besides these, Bechtold and Jacobs (1990) also assume that every break window consists of contiguous planning periods (i.e. no split windows). They noted in their study that assumptions (2), (5), and (7) are consistent with the scheduling practice and are common in the literature, and (3) and (9) can be relaxed by modifying their formulation. With assumptions (4) and (6), however, their formulation would not be appropriate for a shift scheduling problem involving multiple relief and lunch breaks with

unequal duration (e.g. two 15-minute rest breaks and one 30-minute lunch break arrangement found in most service systems) and multiple disjoint break windows associated with them. Furthermore, with assumption (1), their approach is limited to the problems involving shifts starting and ending during the same 24-hour period, and with assumption (8) to the cases not involving any extraordinary break window overlaps. Recognizing these limitations, after discussing the effects of extraordinary break window overlaps, they state in the summary and conclusions section of their study that

> "*Numerous research opportunities exist for the extension of the implicit modeling approach for use in other operating environments. For example, shift scheduling for telephone operators is typically done across 24 hour planning horizons. Moreover, shift scheduling environments exist in practice for which the organization desires to include optimal placement of rest breaks as well as meal breaks in shift assignments* (Hendersen and Berry, 1976)" (Bechtold and Jacobs, 1990, p. 1350).

The study cited by Bechtold and Jacobs (1990) in the above paragraph (Henderson and Berry, 1976), considers a shift scheduling problem involving full-time shifts with two relief breaks and one lunch break with unequal durations and multiple disjoint break windows. Since, this type of shifts are found in most practical applications, we assumed the same scheduling environment to compare the two modeling approaches. Moreover, since the case involving 24-hour continuous operation (cyclical environment) has received considerable attention in practice and in the literature (see, for example, Baker, 1976; Bartoldi, 1981; Bartoldi et al., 1980; Bedworth and Bailey, 1987; Vohra, 1988; Nanda and Browne, 1992, among others), we considered both cyclical and acyclical (less than 24-hour operation) scheduling environments. Solving shift scheduling problems in the scheduling environments considered with the formulation of Bechtold and Jacobs (1990), however, requires the relaxation of assumptions (1), (3), (4), and (6). Below, we present the modifications needed to relax these assumptions. The modifications needed to relax the assumptions concerning the extraordinary break window overlaps (assumption 8) and contiguous break windows, and extend this approach to the general tour scheduling problem (solving daily shift and days-off scheduling together) can be found in Aykin (1995a). Let T be the set of planning periods covered by the shift schedule, and K be the set of shifts including all feasible combinations of different shift types and start times. Define X_k as an integer variable representing the number of employees assigned to shift k, $k \in K$. To model the placement of breaks of a particular type (e.g. first relief break taken before the lunch break) in an environment involving 24-hour continuous operations (acyclical case is discussed later in this section), one needs to distinguish shifts with a break window starting and ending within the same day (for convenience, assume that a work day starts at 7:00 a.m. and ends 7:00 a.m. the following day) from those with a break window starting in one day but continuing into the following day. Consider the first relief break windows for the shifts in K. Let KU1 be the set of shifts with a first relief break window starting and ending within the same day, and KU2 be the set of shifts with a first relief break window starting in one day and continuing into the following day (note that $K \equiv \text{KU1} \cup \text{KU2}$). In the extended formulation, first relief breaks for the shifts in these two sets are scheduled separately by defining separate sets of first relief break variables and introducing two sets of break placement constraints. Let $U1_t$ be an integer variable representing the number of employees assigned to shifts in KU1 and taking their first relief breaks in period t, and $U2_t$ be an integer variable representing the number of employees assigned to a shift in KU2 and taking their first relief break in period t. In this case, first breaks for the shifts in KU1 can be scheduled within their break windows by using the forward pass and backward pass constraints given in Bechtold and Jacobs (1990). In the cyclical scheduling environment considered, a similar set of constraints is needed for the shifts in KU2 in order to schedule correct number of first relief

breaks within their break windows. The number of constraints added for the shifts in KU2 is usually small and is determined by the length of the break window. When an optimal solution to the extended formulation is obtained, break times can be assigned to shifts by applying the simple break scheduling procedure given in Bechtold and Jacobs (1990) to each set separately. Thus, the following two sets of constraints are needed for scheduling the first relief breaks in the extended formulation.

$$\sum_{t\in SU1_j} U1_t - \sum_{k\in TU11_j} X_k \geqslant 0 \quad \text{for all } j \in NU1 - \{qu1\}, \tag{3}$$

$$\sum_{t\in ZU1_j} U1_t - \sum_{k\in TU21_j} X_k \geqslant 0 \quad \text{for all } j \in MU1 - \{pu1\}, \tag{4}$$

$$\sum_{k\in KU1} X_k = \sum_{t\in TU1} U1_t, \tag{5}$$

$$\sum_{t\in SU2_j} U2_t - \sum_{k\in TU12_j} X_k \geqslant 0 \quad \text{for all } j \in NU2 - \{qu2\}, \tag{6}$$

$$\sum_{t\in ZU2_j} U2_t - \sum_{k\in TU22_j} X_k \geqslant 0 \quad \text{for all } j \in MU2 - \{pu2\}, \tag{7}$$

$$\sum_{k\in KU2} X_k = \sum_{t\in TU2} U2_t, \tag{8}$$

where for the first relief breaks for the shifts in KU1, we define

- pu1 earliest first relief break start time for the shifts in KU1
- qu1 latest first relief break start time for the shifts in KU1
- TU1 $\{pu1,\ldots,qu1\}$, the set of all possible first relief break start times for the shifts in KU1
- $SU1_j$ $\{pu1,\ldots,j\}$, the set of all first relief break start times between periods pu1 and j
- $ZU1_j$ $\{j,\ldots,qu1\}$, the set of all first relief break start times between periods j and qu1
- NU1 set of latest first relief break start times (in ascending order) for the shifts in KU1
- MU1 set of earliest first relief break start times (in ascending order) for the shifts in KU1,
- $TU11_j$ the set of shifts in KU1 with a first relief break window lying completely between periods pu1 and j, and
- $TU21_j$ the set of shifts in KU1 with a first relief break window lying completely between periods j and qu1

In a similar fashion, for the shifts in KU2, we define the parameters and the sets pu2, qu2, TU2, $SU2_j$, $ZU2_j$, NU2, MU2, $TU12_j$, and $TU22_j$.

To model lunch break placements correctly, define, similar to KU1 and KU2 defined for the first relief breaks above, the sets KW1 and KW2 ($K \equiv KW1 \cup KW2$). Also define the integer variables $W1_t$ and $W2_t$ as the number of employees assigned, respectively, to shifts in sets KW1 and KW2 and starting their lunch breaks in period t. The following two sets of constraints are needed to schedule lunch breaks within the correct break windows for the shifts in KW1 and KW2.

$$\sum_{t\in SW1_j} W1_t - \sum_{k\in TW11_j} X_k \geqslant 0 \quad \text{for all } j \in NW1 - \{qw1\}, \tag{9}$$

$$\sum_{t\in ZW1_j} W1_t - \sum_{k\in TW21_j} X_k \geqslant 0 \quad \text{for all } j \in MW1 - \{pw1\}, \tag{10}$$

$$\sum_{k\in KW1} X_k = \sum_{t\in TW1} W1_t, \tag{11}$$

$$\sum_{t\in SW2_j} W2_t - \sum_{k\in TW12_j} X_k \geqslant 0 \quad \text{for all } j \in NW2 - \{qw2\}, \tag{12}$$

$$\sum_{t\in ZW2_j} W2_t - \sum_{k\in TW22_j} X_k \geqslant 0 \quad \text{for all } j \in MW2 - \{pw2\}, \tag{13}$$

$$\sum_{k\in KW2} X_k = \sum_{t\in TW2} W2_t, \tag{14}$$

where for the lunch breaks for the shifts in KW1, we define

pw1	earliest lunch break start time for the shifts in KW1
qw1	latest lunch break start time for the shifts in KW1
TW1	$\{pw1,\ldots,qw1\}$, the set of all possible lunch break start times for the shifts in KW1
$SW1_j$	$\{pw1,\ldots,j\}$, the set of all lunch break start times between periods pw1 and j
$ZW1_j$	$\{j,\ldots,qw1\}$, the set of all lunch break start times between periods j and qw1
NW1	set of latest lunch break start times (in ascending order) for the shifts in KW1
MW1	the set of earliest lunch break start times (in ascending order) for the shifts in KW1
$TW11_j$	set of shifts in KW1 with a lunch break window lying completely between periods pw1 and j, and
$TW21_j$	the set of shifts in KW1 with a lunch break window lying completely between periods j and qw1

Similarly, define pw2, qw2, TW2, $SW2_j$, $ZW2_j$, NW2, MW2, $TW12_j$, and $TW22_j$ for the lunch breaks for the shifts in KW2.

Above, constraints (9)–(11) and (12)–(14) ensure that two sets of lunch break times are scheduled for the shifts in KW1 and KW2 separately.

To schedule the second relief breaks, define the sets KV1 and KV2 ($K \equiv KV1 \cup KV2$). Also define the integer variables $V1_t$ and $V2_t$ as the number of employees assigned, respectively, to shifts in KV1 and KV2 and taking their second relief breaks in period t. The following two sets of constraints are needed to schedule second relief breaks within the correct break windows for the shifts in KV1 and KV2.

$$\sum_{t\in SV1_j} V1_t - \sum_{k\in TV11_j} X_k \geqslant 0 \quad \text{for all } j \in NV1 - \{qv1\}, \tag{15}$$

$$\sum_{t\in ZV1_j} V1_t - \sum_{k\in TV21_j} X_k \geqslant 0 \quad \text{for all } j \in MV1 - \{pv1\}, \tag{16}$$

$$\sum_{k\in KV1} X_k = \sum_{t\in TV1} V1_t, \tag{17}$$

$$\sum_{t\in SV2_j} V2_t - \sum_{k\in TV12_j} X_k \geqslant 0 \quad \text{for all } j \in NV2 - \{qv2\}, \tag{18}$$

$$\sum_{t\in ZV2_j} V2_t - \sum_{k\in TV22_j} X_k \geqslant 0 \quad \text{for all } j \in MV2 - \{pv2\}, \tag{19}$$

$$\sum_{k\in KV2} X_k = \sum_{t\in TV2} V2_t, \tag{20}$$

where for the second relief breaks for the shifts in KV1, we define

pv1	earliest second relief break start time for the shifts in KV1
qv1	latest second relief break start time for the shifts in KV1
TV1	$\{pv1,\ldots,qv1\}$, the set of all possible second relief break start times for the shifts in KV1
$SV1_j$	$\{pv1,\ldots,j\}$, the set of all second relief break start times between periods pv1 and j
$ZV1_j$	$\{j,\ldots,qv1\}$, the set of all second relief break start times between periods j and qv1
NV1	set of latest second relief break start times (in ascending order) for the shifts in KV1

MV1 set of earliest second relief break start times (in ascending order) for the shifts in KV1

$TV11_j$ set of shifts in KV1 with a second relief break window lying completely between periods pv1 and j, and

$TV21_j$ set of shifts in KV1 with a second relief break window lying completely between periods j and qv1

Similarly, define pv2, qv2, TV2, $SV2_j$, $ZV2_j$, NV2, MV2, $TV12_j$, and $TV22_j$ for the second relief breaks for the shifts in KV2.

Constraints (15)–(17) and (18)–(20) above ensure that two sets of second relief are scheduled within the break windows specified for the shifts in KV1 and KV2 separately.

We now present the extended formulation of Bechtold and Jacobs for a cyclical case involving multiple rest and lunch breaks, and disjoint break windows. Let c_k be the cost of assigning an employee to shift k, $k \in K$, a_{kt} be equal to one if period t is within the workspan (a work or a break period) of shift k, and zero otherwise, and b_t be the number of employees needed in period t. Then, the extended formulation can be stated as follows:

$$\text{Minimize} \quad \sum_{k \in K} c_k X_k \tag{21}$$

subject to

$$\sum_{k \in K} a_{kt} X_k - \sum_{j \in TU1} v1_{jt} U1_j - \sum_{j \in TW1} \omega 1_{jt} W1_j - \sum_{j \in TV1} v1_{jt} V1_j - \sum_{j \in TU2} v2_{jt} U2_j - \sum_{j \in TW2} \omega 2_{jt} W2_j - \sum_{j \in TV2} v2_{jt} V2_j \geq b_t$$

for all $t \in T$,

constraints (3)–(20), (22)

$X_k, U1_t, U2_t, W1_t, W2_t, V1_t, V2_t \geq 0$ and integer.

where

$v1_{jt}$ 1 if period t is a first relief break period for the employees starting their breaks in period j, $j \in TU1$, and zero otherwise,

$\omega 1_{jt}$ 1 if period t is a lunch break period for the employees starting their breaks in period j, $j \in TW1$, and zero otherwise,

$v1_{jt}$ 1 if period t is a second relief break period for the employees starting their breaks in period j, $j \in TV1$, and zero otherwise,

and $v2_{jt}, \omega 2_{jt}$, and $v2_{jt}$ are defined similarly for the shifts in TU2, TW2, and TV2, respectively.

In the above formulation, constraint (22) ensures that the number of employees assigned to various shifts and available in period t minus the number of employees taking their first relief, lunch, and second relief breaks is sufficient to meet demand and provide the desired level of service in that period.

The equality constraints (5), (8), (11), (14), (17), and (20) may be substituted in the associated break constraints in a manner similar to Bechtold and Jacobs (1990) to reduce the number of nonzeros whenever appropriate. Our preliminary tests with this reduced form, however, did not show any computational advantage over the basic model.

Note that if the scheduling problem is an acyclical one involving multiple relief and lunch breaks and break windows, then $KU1 \equiv K$, $KW1 \equiv K$, and $KV1 \equiv K$, with $KU2 \equiv KW2 \equiv KV2 = \emptyset$. Hence, variables $U2_t$, $W2_t$, and $V2_t$, and constraints (6)–(8), (12)–(14), and (18)–(20) are not needed in the above formulation.

Recently, Thompson (1995) applied the formulation of Bechtold and Jacobs together with the implicit representation of shifts described in Moondra (1976) in a shift scheduling environment involving a high degree of shift length and start time flexibility. Given a set of shift start times and a set of shift lengths associated with every shift start time (e.g. shifts with lengths 6, 6.5, ..., 9.5, and 10 hours are assumed to be available to start every 30 minutes throughout the day), Moondra's approach was used to model shifts with varying lengths and start times, and the modeling approach of Bechtold and Jacobs to model break

placements implicitly to reduce the size of the formulation. The extension presented in Thompson (1995), however, has a number of limitations. First, it assumes that a high degree of flexibility in terms of both the shift start times and shift lengths is available. In practice, usually the company, union, and legal requirements are followed in determining shift start times together with shift lengths (see, for example, Segal, 1974; Henderson and Berry, 1976; Buffa, 1980; Bedworth and Bailey, 1987; Love and Hoey, 1990; Nanda and Browne, 1992; Spencer et al., 1992). Hence, except for such systems as fast food restaurants, the degree of combined shift length and shift start time flexibility assumed is not common in practice. If a high degree of shift length flexibility is not available with *most* of the shift start times, on the other hand, the formulation presented in Thompson (1995) would offer little advantage over the formulation of Bechtold and Jacobs. For instance, in a problem involving only full-time employees (e.g. Henderson and Berry, 1976), or in a problem with alternative shift lengths that are available only at certain shift start times (e.g. Segal, 1974), the formulation given in Thompson would offer little (if any) advantage over the formulation of Bechtold and Jacobs. Another limitation of the formulation given in Thompson (1995) is that the break windows for a shift type are defined with reference to the shift start and finish times by specifying the minimum and maximum pre- and post-break work periods. As a result, even when two shifts share similar attributes (e.g. same break duration, even the same start time), the length of the break window decreases with the shift length. In the illustrative example given in Thompson (1995, Section 2.2.), for example, while the break window is 3 hours long for a 7-hour shift, the length of the break window decreases to 2 hours for a 6-hour shift, and to 1 hour (that is, a "fixed" meal break time for a 1-hour meal break) for a 5-hour shift even when they all start at the same time. In practice, start time and length of a shift determine the number and type of breaks (e.g. rest, lunch, idle period for a split-shift) employees receive. Together with them, break windows are specified with their lengths and position with respect to some other shift features (e.g. start time, "ideal" break time, etc). As reported extensively in the literature (see, for example, Segal, 1974; Henderson and Berry, 1976; Bedworth and Bailey, 1987; Bechtold and Jacobs, 1990; Nanda and Browne, 1992; Jarrah et al., 1994), the length and positioning of a break window usually do not change with the shift start time or length. This way the degree of break scheduling flexibility provided by break windows remains the same for all shift types to better match demand, improve manpower utilization, and provide timely relief to employees. This type of scheduling environment cannot be modeled efficiently with the formulation described in Thompson (1995). Also, the minimum shift length for a particular shift type, and the minimum pre- and post-break work periods should be examined carefully to avoid infeasibility.

Another limitation of the formulation presented by Thompson is its inability to model a realistic cost structure accurately. The formulation presented assumes that the cost of assigning an employee to a shift is a linear function of only its duration and not whether it is a full-time or a part-time shift, its start time or the time periods it covers (e.g. day shift vs. night shift). As also stated in Thompson (1995), to model a realistic cost structure accurately, a formulation with explicit shift representation is needed. And, finally, like the formulation of Bechtold and Jacobs, the formulation given in Thompson is limited to the cases covering less than 24 hours (i.e. acyclical problem). In summary, the formulation presented in Thompson (1995) is applicable to special cases satisfying these conditions and cannot be applied to the general shift scheduling problem in its present form. Therefore, this extension of Bechtold and Jacobs (1990) is not considered in this study.

2.3. Implicit formulation of Aykin (1996)

In a recent study (Aykin, 1996), we presented another implicit formulation for the general shift scheduling problem. Like Bechtold and Jacobs (1990), Aykin (1996) is modeling the break placements implicitly. The approaches used in these two studies, however, are different in the way the breaks times are matched with the shifts. While the

formulation given by Bechtold and Jacobs (1990) uses break variables associated with each planning period and matches breaks with the shifts implicitly, the formulation given in Aykin (1996) defines separate break variables for each shift. This approach extends the implicit break representation idea introduced in Gaballa and Pearce (1979) to the general case involving multiple rest and lunch breaks and multiple disjoint break windows. In the special case considered by Gaballa and Pearce, this approach required more variables and constraints than the equivalent set covering formulation. Therefore, their approach has not been generalized or considered any further in the literature. Aykin (1996) showed that although apparently not beneficial in the special case considered by Gaballa and Pearce (1979), this approach is surprisingly very successful in reducing the number of variables needed in the general shift scheduling problem. The number of nonzeros and the density of the A-matrix are also much lower than the set covering formulation. Besides these, the new formulation makes the same assumptions with the set covering formulation and is applicable to all cases that the set covering approach is applicable.

To formulate the problem using this approach, we define K as the set of all shifts including all feasible combinations of shift start times and lengths, and X_k as the number of employees assigned to shift k. Let U_{kt}, W_{kt}, and V_{kt} be the break variables representing the number of employees assigned to shift k and taking their first relief, lunch, and second relief breaks in period t, respectively. Let $B1_k$, BL_k, and $B2_k$ be the sets of planning periods forming the first relief, lunch, and second relief break windows of shift k, respectively. In this formulation, the first relief break variable U_{kt} is defined only for $t \in B1_k$, lunch break variable W_{kt} for $t \in BL_k$, and the second relief break variable V_{kt} for $t \in B2_k$. Further, define $T1_t$, TL_t, and $T2_t$ as the sets of shifts for which period t is a break start time within the break windows for the first relief, lunch, and second relief break, respectively. Also, let a_{kt} be one if period t is in the shift span (a work or a break period during the shift) of shift k and zero otherwise. The general shift scheduling problem can then be formulated as follows:

$$\text{Minimize} \quad \sum_{k \in K} c_k X_k \tag{23}$$

subject to

$$\sum_{k \in K} a_{kt} X_k - \sum_{k \in T1_t} U_{kt} - \sum_{k \in TL(t-1)} W_{k(t-1)} - \sum_{k \in TL_t} W_{kt} - \sum_{k \in T2_t} V_{kt} \geqslant b_t \quad \text{for all } t \in T, \tag{24}$$

$$X_t - \sum_{t \in B1_k} U_{kt} = 0 \quad \text{for all } k \in K, \tag{25}$$

$$X_t - \sum_{t \in BL_k} W_{kt} = 0 \quad \text{for all } k \in K, \tag{26}$$

$$X_t - \sum_{t \in B2_k} V_{kt} = 0 \quad \text{for all } k \in K, \tag{27}$$

$X_t, U_{kt}, W_{kt}, V_{kt} \geqslant 0$ and integer.

In the above formulation, break windows are specified by $B1_k$, BL_k, and $B2_k$. In the extended formulation of Bechtold and Jacobs presented before, break windows determine the earliest and latest break start times (i.e. pu1, qu1, etc.) and the break placement requirements in the break constraints. Except for the no-extraordinary break window overlaps and contiguous break window requirements of Bechtold and Jacobs (1990) (both are relaxed in Aykin, 1995a,b), the extended formulation of Bechtold and Jacobs given above and the implicit modeling approach given by Aykin are equivalent to the set covering formulation.

Also note that the number of variables and the number of constraints in the above formulation can be reduced by substituting one of the break constraints (25)–(27) associated with each shift in the other break constraints and the demand constraints (24). Our tests with this reduced form, however, did not show any computational advantage over the basic model in terms of model reliability and solution time.

3. Experimental evaluation

In this section, we describe the shift scheduling environment considered in the computational experiments, discuss the characteristics of the implicit formulations, and report our results with 220 problems.

3.1. Scheduling environment

We varied the scheduling environment considered in the test problems along four dimensions: relief and lunch break window sizes, shift start time pattern, cyclical-acyclical operations, and demand pattern. Employee demand is assumed to be determined for 96 quarter-hour ($= 4 \times 24$ hours) planning periods, $T = \{1, \ldots, 96\}$. Concerning the shift start times, we considered two schemes; (i) 9-hour shifts starting on the hour or half-hour, and (ii) 9-hour shifts starting every 15 minute. Each employee is assumed to be given one 30-minute lunch break and two 15-minute relief breaks (one before and one after the lunch break). We specify the break windows with reference to "ideal" break times as follows: the ideal first relief break time for a shift is specified as 2 hours after the start of the shift, the ideal lunch start time is 4 hours and 15 minutes (4 hours of work plus the first relief break) after the start of the shift, and the ideal second relief break time is 6 hours and 45 minutes (6 hours of work plus the first relief and lunch breaks) after the start of the shift. The break windows are specified by setting the earliest break start times half an hour earlier than the ideal break start times. To test the modeling approaches in situations with varying difficulty level, we considered 11 break window length combinations allowing between 4 and 7 break start times for the relief breaks (1 hour to 1 hour and 45 minutes windows) and between 3 and 7 lunch break start times (1–2 hour windows). Together with these, we considered five demand patterns (Fig. 1). The following three demand patterns were reported in the literature and were either obtained from or resembling real patterns from service systems: (i) the trimodal demand profile given in Segal (1974) for telephone operators with a minimum quarter hour demand of 2 and a maximum of 89, (ii) the bimodal demand profile with a minimum quarter hour demand of 2 and a maximum of 9 reported by Henderson and Berry (1976) for the General Telephone and Electronics Company of California, and (iii) the unimodal demand profile with a minimum quarter hour demand of 2 and a maximum of 27 given in Buffa et al. (1976) for the General Telephone Company of California. The two other demand patterns were synthetically generated; a bimodal and a trimodal sinusoidal demand pattern with a minimum quarter hour demand of 5 and a maximum of 50. We generated a total of 220 problems by considering both cyclical and acyclical versions of the combinations of these factors and their levels.

3.2. Model size characteristics

The number of variables, number of constraints, number of nonzeros and the density of the A-matrix for the implicit formulations of Bechtold and Jacobs (1990), Aykin (1996), and the set covering formulation are shown for some representative cases (from the smallest to the largest considered in this study) in Tables 1 and 2, respectively. In these tables, n_K is the number of shift start times in K, and $n1_k$, $n2_k$, and nL_k show the number of break start times in the first relief, second relief, and lunch break windows, respectively. As can be seen from Table 1, in the acyclical problems considered, the number of variables (that is, the number of shift variations) in the set covering model varies between 1984 and 20 923. In contrast, the extended implicit modeling approach of Bechtold and Jacobs requires between 223 and 262, and the implicit modeling approach of Aykin requires between 403 and 1342 variables. In every case, the extended formulation of Bechtold and Jacobs requires the smallest number of variables. Aykin's approach requires more than the extended formulation of Bechtold and Jacobs but significantly less than the set covering formulation. The formulation of Aykin, on the other hand, has a substantially smaller number of nonzeros in the A-matrix than the other two approaches; while the set covering formulation

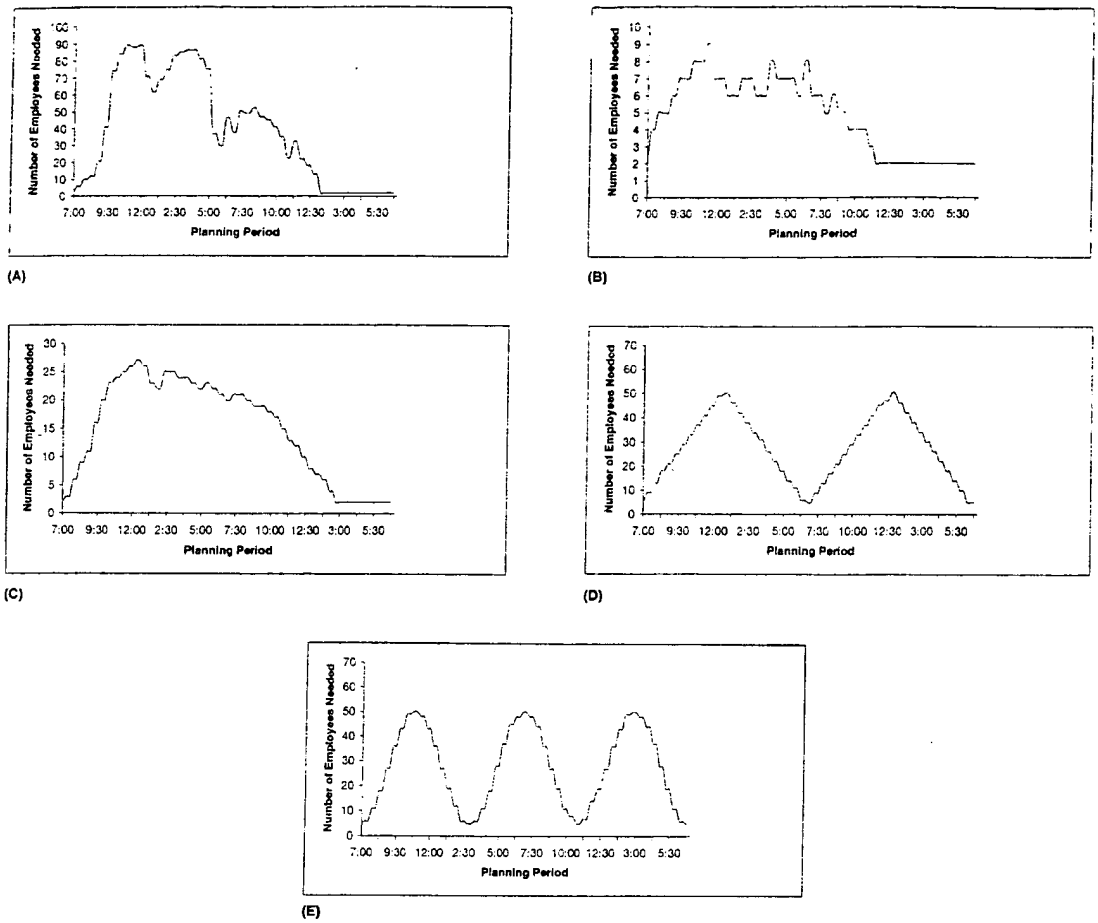


Fig. 1. Demand patterns: (A) Segal (1974), (B) Henderson and Berry (1976), (C) Buffa et al. (1976), (D) bimodal demand pattern, and (C) trimodal demand pattern.

has between 63 488 and 1 053 696 nonzeros and the formulation of Bechtold and Jacobs has between 10 387 and 56 124 nonzeros in the A-matrix, the formulation of Aykin (1995a,b) has between 2077 and 8448 nonzeros (i.e. about 80% less than the equivalent formulation of Bechtold and Jacobs, and 96.7–99.2% less than the equivalent set covering formulation). Furthermore, while the density of the A-matrix of Aykin's formulation varies between 1.5% and 2.9%, it is between 16.8% and 22.4% for the A-matrix of the extended formulation of Bechtold and Jacobs, and 33.3% for the set covering formulation. Note that since, in the set covering model, all shift variations are enumerated, only demand constraints are needed. As can be seen from Table 1, the implicit modeling approaches require more constraints in order to assure correct break placements. Similar conclusions can be reached from the results presented in Table 2 for the cyclical problems.

As discussed before, the implicit formulations of this problem are taking advantage of the fact that only some of the information about the break times generated by the set covering formulation is needed in the model to obtain an optimal schedule. This is true since, in general, the shift scheduling problem has a large number of alternative optimal solutions. In most cases, given an optimal solution, one can generate alternative optimal schedules by

Table 1
Problem characteristics: acyclical problems

n_K	$n_{1k} = n_{2k}$	n_{Lk}	Number of variables			Number of constraints			Number of nonzeros in A-matrix			% Nonzeros in A-matrix		
			Set covering	Bechtold and Jacobs	Aykin	Set covering	Bechtold and Jacobs	Aykin	Set covering	Bechtold and Jacobs	Aykin	Set covering	Bechtold and Jacobs	Aykin
31	4	4	1984	223	403	96	279	189	63488	10387	2077	33.3	16.8	2.9
	5	3	2325	224	434	96	279	189	74400	10448	2108	33.3	16.8	2.7
	6	5	5580	228	558	96	279	189	178560	10698	2418	33.3	16.9	2.4
	7	7	10638	232	682	96	279	189	340256	10964	2728	33.3	17.0	2.2
61	4	4	3904	253	793	96	459	279	124928	25867	4087	33.3	22.3	1.9
	5	3	4575	254	854	96	459	279	146400	25988	4148	33.3	22.3	1.8
	6	5	10980	258	1098	96	459	279	351360	26478	4758	33.3	22.4	1.6
	7	7	20923	262	1342	96	459	279	669536	26968	5368	33.3	22.4	1.5

Table 2
Problem characteristics: cyclical problems

n_K	$n_{1k} = n_{2k}$	n_{Lk}	Number of variables			Number of constraints			Number of nonzeros in A-matrix			% Nonzeros in A-matrix		
			Set covering	Bechtold and Jacobs	Aykin	Set covering	Bechtold and Jacobs	Aykin	Set covering	Bechtold and Jacobs	Aykin	Set covering	Bechtold and Jacobs	Aykin
48	4	4	3072	348	624	96	378	240	98304	22042	3216	33.3	16.8	2.1
	5	3	3600	350	672	96	378	240	115200	22136	3264	33.3	16.7	2.0
	6	5	8640	358	864	96	378	240	276480	21996	3744	33.3	16.3	1.8
	7	7	16464	378	1056	96	378	240	526848	21868	4224	33.3	15.8	1.7
96	4	4	6144	402	1248	96	666	384	196608	57522	6432	33.3	21.5	1.4
	5	3	7208	404	1344	96	666	384	230400	57364	6528	33.3	21.3	1.3
	6	5	17280	412	1728	96	666	384	552960	56720	7488	33.3	20.7	1.2
	7	7	32928	420	2112	96	666	384	1053696	56124	8448	33.3	20.1	1.1

Table 3
Summary of computational experiments

	Bechtold and Jacobs	Aykin
Total number of problems considered	220	220
Number of problems remained unsolved		
After 1st pass	45	15
After 2nd pass	32	2
Number of problems remained unsolved with both approaches		
After 1st pass	11	11
After 2nd pass	1	1
Number of problems with same CPU time[a]	4	4
Number of problems with a better CPU time	12 (5.5%)	203 (92.3%)
Average[b] CPU time (s) per problem	331.36 (188)	113.04 (218)
Lower bound on CPU time (s) per problem	504.85	200.53

[a] The difference between the solution times $\leqslant 0.5$ s.
[b] After two passes.

switching the breaks of different shifts that have overlapping break windows, provided this does not violate any of the demand or break constraints. Consequently, the set covering, and the implicit modeling approaches are three possible strategies to solve this problem; as the set covering formulation requires, one may enumerate all rest and work patterns and determine only one of these optimal solutions that provides a complete schedule or, as in the extended formulation of Bechtold and Jacobs (1990), an optimal solution specifies only the active shifts and sufficient number of breaks,and a complete schedule is obtained from this solution by postprocessing. As a third strategy, the implicit formulation of Aykin (1996) is providing an approach that is taking advantage of the fact that one can formulate the problem to obtain slightly more information in the optimal solution than Bechtold and Jacobs (1990) (but still less than the set covering formulation) by scheduling shifts and breaks associated with each shift separately. Thus, the formulation of Bechtold and Jacobs (1990) is more compact in terms of the number of variables. The primary advantage of Aykin (1996), on the other hand, is that it requires significantly smaller number of constraints and nonzeros in the A-matrix than Bechtold and Jacobs (1990). As discussed in the computational results section, these approaches are significantly different in the amount of computational effort needed and reliability.

3.3. Experimental results

All 220 problems were formulated using the implicit modeling approaches of Bechtold and Jacobs (1990) and Aykin (1996), and were solved using LINDO (Version 5.3.) together with a FORTRAN interface that generates and inputs problem data. All runs were completed in batch mode on a Sun Sparc 512 computer. Since the superiority of both implicit modeling approaches to the set covering formulation has been demonstrated separately with computational experiments in Bechtold and Jacobs (1990), and Aykin (1998), the set covering formulation was not considered in our computation experiments. We also note that the memory requirements (e.g. number of variables) for the set covering formulation of most of the problems considered in our study would exceed the limits of the software/hardware combination used.

Both Bechtold and Jacobs (1991) and Aykin (1996) reported that the break variables in their formulations tend to be integer when the shift variables are integer. Therefore, a branching strategy giving higher priority to the shift variables X_k was implemented in LINDO. The CPU time limit was set to 1200 seconds. No attempt was made to determine the branching requirements and total solution time if LINDO could not locate an integer optimal solution to a problem within the CPU time limit. As seen in Table 3, this was the case in 45 (20.5%) out of 220 problems with the

Table 4
Computational results

Number of shift start times (n_K)	Number of shift variations			Bechtold and Jacobs				Aykin			
	Min.	Avg.	Max.	Solution time[a]			Number of prob. remain unsolved	Solution time[a]			Number of prob. remain unsolved
				Min.	Avg.	Max.		Min.	Avg.	Max.	
31	1984	4856	10638	3.77	171.01 (292.59)	1231.25	3	3.16	97.44 (147.4)	1226.30	0
48	3072	7519	16464	48.88	274.76 (313.40)	1342.58	1	5.20	98.20 (96.86)	1224.73	0
61	3904	9555	20923	19.46	244.17 (438.76)	1437.14	6	5.63	119.02 (231.84)	1414.82	1
96	6144	15037	32928	165.93	806.14 (974.66)	2376.66	22	16.32	138.06 (326.03)	1419.65	1

[a] CPU seconds.

extended formulation of Bechtold and Jacobs, and in 15 (6.8%) out of 220 problems with the approach of Aykin. These cases include the 11 problems in which both approaches failed to locate an optimal integer solution within the allowed time. In order to gain more information, we made a second attempt using a branching strategy giving a higher priority to the variables associated with the shifts covering higher total demand during its shift span. In case of a tie, the variables associated with the shifts with an earlier start time were chosen for branching. If a second attempt was made, the total time for both passes is reported. Thus, the CPU time limit for two passes is 2400 seconds. Out of 45 problems which remained unsolved after the first pass with the approach of Bechtold and Jacobs, 13 more solved in the second pass. With the approach of Aykin, on the other hand, out of 15 not solved during the first pass, only 2 problems remained unsolved after the second pass. In summary, after two passes, 32 problems with the extended formulation of Bechtold and Jacobs, or about 14.5% of the problems, and 2 problems with the formulation of Aykin, or about 0.9%, remained unsolved within the allotted time.

As can be seen from Table 3, in 203 (92.3%) of the problems, the formulation given in Aykin (1996) required less CPU time to locate an optimal integer solution than the extended formulation of Bechtold and Jacobs (1990). In 12 problems, the extended formulation of Bechtold and Jacobs (1990) required less CPU time than that of Aykin (1996). And in 4 of the remaining 5 problems (out of 220), the difference between the solution times with the two formulations was less than 0.5 seconds. Hence, these cases were reported as requiring the same amount of CPU time. Only one case remained unsolved with both formulations.

Out of 220 problems, 187 of them were solved optimally by both formulations. The average CPU time for these problems with the extended formulation of Bechtold and Jacobs is 331.36 seconds. In contrast, for the same 187 problems, the average CPU time with the formulation of Aykin is 113.04, about 1/3 of the average time needed with the formulation of Bechtold and Jacobs. Table 4 provides more detailed information concerning the solution times. In this table, the minimum, aver-

age, and maximum number of shifts, and minimum, average, and maximum CPU times with both formulations for the problems in each group are also presented. As seen from these results, the average solution time increases with both formulations as the number of shifts increases. The rate of increase with the formulation of Bechtold and Jacobs is, however, significantly higher than with the formulation of Aykin. Also note that the average solution time needed with the extended formulation of Bechtold and Jacobs in these problem groups is between 1.76 and 5.84 times higher than that with the formulation of Aykin. Table 4 also shows the number of problems remained unsolved (after two passes) in each group. For each group, below the average solution time, we report (in parenthesis) the average CPU time spent per problem in the group to include the unsolved

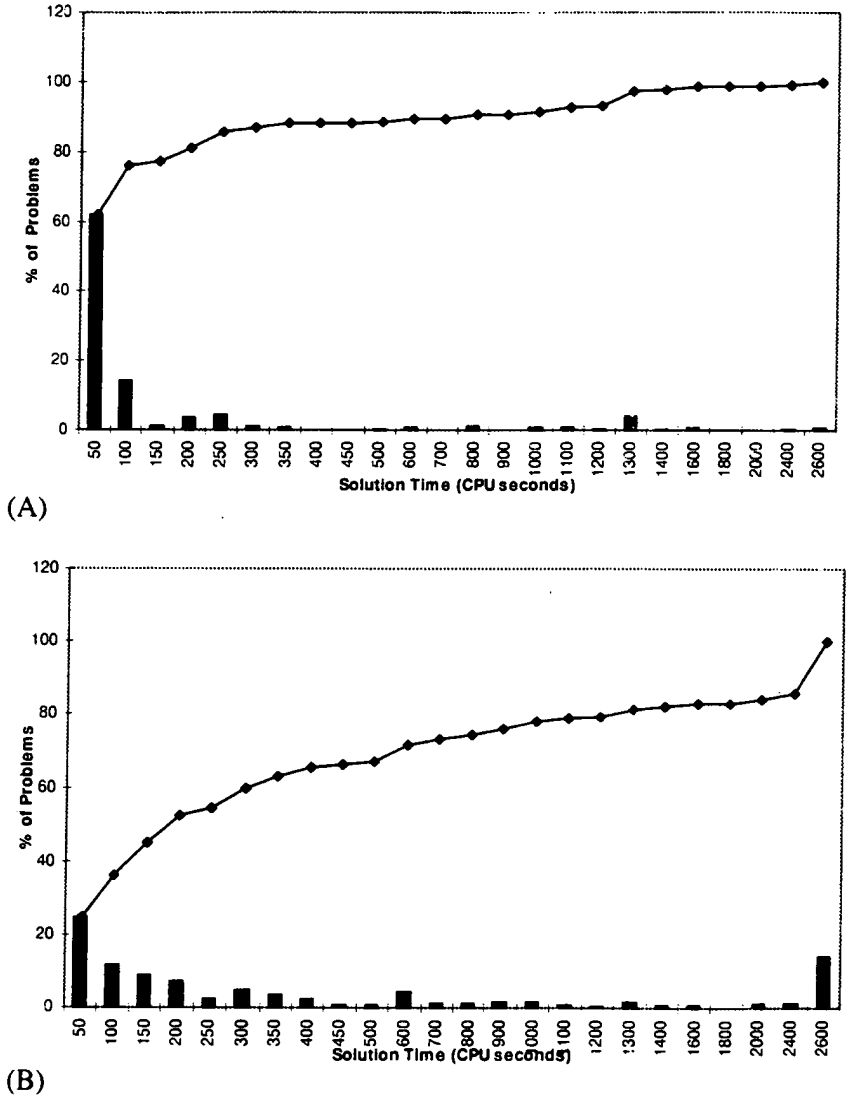


Fig. 2. Distribution of the solution times: (A) with the formulation of Aykin (1996), (B) with the formulation of Bechtold and Jacobs (1990).

problems. These averages were obtained by including 2400 seconds spent for each case that is remaining unsolved after two passes together with the actual solution times for the problem that were solved optimally. These average times provide a lower bound on the average solution times per problem. As can be seen, the lower bounds for the formulation of Bechtold and Jacobs are also higher than the corresponding lower bounds for the formulation of Aykin.

Finally, Fig. 2 shows the distribution of the individual solution times. In this figure, the histogram shows the percentage of the problems with a solution time in the given interval, and the curve shows the cumulative percentage of the problems versus the solution time. The dashed lines mark the points where the scale of the solution time axis is changed due to the spread of the recorded solution times. As can be seen from this figure, in more than 60% of the problems, an optimal integer solution was obtained within the first 50 seconds with the formulation of Aykin and, in more than 75% of the problems, within 100 seconds. Also note that about 15% of the problems required between 100 and 500 seconds and only 10% required more than 500 seconds. In contrast, with the extended formulation of Bechtold and Jacobs, only about 25% of the problems were solved within the first 50 seconds, and about 37% within the first 100 seconds. About 30% of the problems required between 100 and 500 seconds, and 33% required more than 500 seconds. This approach requires more than 2400 seconds in 14.5% of the problems that remained unsolved.

As discussed in Section 3.2, in terms of the number of constraints, the extended formulation of Bechtold and Jacobs (1990) is a more compact formulation of the problem than Aykin (1996). However, the results presented in this section show that this is not providing any computational advantage over Aykin (1996). A comparison of the optimal LP objective values revealed that both formulations are producing the same LP objective values. However, in most cases (as reported in Table 3, 92.3%), solving (i.e. pivoting) the initial LP relaxation as well as the subsequent branch subproblems consume significantly more time with Bechtold and Jacobs (1990) than Aykin (1996). One reason for this may be the substantially higher density of the A-matrix of the Bechtold and Jacobs formulation. In summary, these results show that the formulation given in Aykin (1996) offers better modeling reliability, and requires less memory and, on average, one-third of the time needed with the formulation of Bechtold and Jacobs (1990).

4. Summary

In this paper, we experimentally compared the extended formulation of Bechtold and Jacobs (1990) and the implicit modeling approach of Aykin (1996) to the general shift scheduling problem through solving 220 problems. We considered two criteria; model reliability and solution time. In terms of model reliability, while 32 (14.5%) out of 220 problems remained unsolved after two passes with the extended formulation of Bechtold and Jacobs, only 2 (0.9%) of the problems remained unsolved with the formulation of Aykin. Further, the average solution time for the problems solved by both approaches was 331.36 seconds with the extended formulation of Bechtold and Jacobs, and 113.04 seconds with the formulation of Aykin. In summary, our results show that the formulation given in Aykin (1996), although requiring more variables than the equivalent formulation of Bechtold and Jacobs, requires smaller number of constraints, has lower density and smaller number of nonzeros in the A-matrix, offers better model reliability, and requires on average one-third of the time needed with the extended formulation of Bechtold and Jacobs. The fact that these results were obtained with a problem set including the largest shift scheduling problems solved optimally in the literature using an off-the-shelf integer programming software is very encouraging in terms of the practical applications of the implicit scheduling models in service systems.

References

Aykin, T., 1998. A composite branch and cut algorithm for optimal shift scheduling with multiple breaks and break windows. Journal of the Operations Research Society 48, 1–13.

Aykin, T., 1996. Optimal shift scheduling with multiple break windows. Management Science 42, 591–603.

Aykin, T., 1995a. A generalized implicit modeling approach for the shift and tour scheduling problems. GSM Working Paper #95-29, Graduate School of Management, Rutgers University, Newark, NJ.

Aykin, T., 1995b. Reformulating the tour scheduling problem to improve solvability. GSM Working Paper #95-35, Graduate School of Management, Rutgers University, Newark, NJ.

Baker, K., 1976. Workforce allocation in cyclical scheduling problems: A survey. Operational Research Q 27, 155–167.

Bartoldi, J.J., 1981. A guaranteed-accuracy round-off algorithm for cyclic scheduling and set covering. Operations Research 29, 501–510.

Bartoldi, J.J., Orlin, J.B., Ratliff, H.D., 1980. Cyclic scheduling via integer programs with circular ones. Operations Research 29, 1074–1085.

Bechtold, S.E., Jacobs, L.W., 1990. Implicit modeling of flexible break assignments in optimal shift scheduling. Management Science 36 (11), 1339–1351.

Bedworth, D.D., Bailey, J.E., 1987. Integrated Production Control Systems, second ed. Wiley, New York.

Buffa, E.S., 1980. Modern Production and Operations Management, sixth ed. Wiley, New York.

Dantzig, G.B., 1954. A comment on Edie's "Traffic delays at toll booths". Operations Research 2 (3), 339–341.

Edie, L.C., 1954. Traffic delays at toll booths. Operations Research 2 (2), 107–138.

Gaballa, A., Pearce, W., 1979. Telephone sales manpower planning at Qantas. Interfaces 9 (3), 1–9.

Henderson, W.B., Berry, W.L., 1976. Heuristic methods for telephone operator shift scheduling: An experimental analysis. Management Science 22 (12), 1372–1380.

Jarrah, A.I.Z., Bard, J.F., deSilva, A.H., 1994. Solving large-scale tour scheduling problems. Management Science 40, 1124–1144.

Love, R.R., Hoey, J., 1990. Management science improves fast-food operations. Interfaces 20, 21–29.

Moondra, S.L., 1976. An L.P model for work force scheduling for Banks. Journal of Bank Research 6, 299–301.

Nanda, R., Browne, J., 1992. Introduction to Employee Scheduling. Van Nostrand Reinhold, New York.

Segal, M., 1974. The operator-scheduling problem: A network flow approach. Operations Research 22, 808–823.

Spencer III, T., Brigandi, T., Barlow, G., Gomez, R., 1992. A decision support system for workforce management. Modelling and Simulation 23 (5).

Thompson, G.M., 1995. Improved implicit optimal modeling of the labor shift scheduling problem. Management Science 41, 595–607.

Vohra, R.V., 1988. A quick heuristic for some cyclic staffing problems with breaks. Journal of the Operational Research Society 39, 1057–1061.

Journal of the Operational Research Society (1998) 49, 603–615
http://www.stockton-press.co.uk/jor

A composite branch and cut algorithm for optimal shift scheduling with multiple breaks and break windows

T Aykin
Bell Laboratories, New Jersey, USA

This paper presents a new branch and cut algorithm for optimal shift scheduling with multiple breaks and break windows. The proposed algorithm is based on an implicit formulation of the problem requiring significantly smaller number of variables than the set covering formulation of Dantzig. The new algorithm, adding cuts, developing upper bounds for the variables, and employing an efficient rounding heuristic, was tested successfully with 90 test problems involving between 2160 and 32 928 shift variations and five demand profiles. Our results show that large shift scheduling problems can be solved optimally with the proposed branch and cut algorithm and the implicit formulation.

Keywords: shift scheduling; service operations management; integer programming

Introduction

The shift scheduling problem arises in such situations as scheduling telephone operators, agents at telemarketing centres, and attendants at highway toll booths, etc. The problem involves determining the number of employees to be assigned to various shifts (specified by shift start time, shift length, number and type of breaks, etc) and the timing of their relief and lunch breaks so as to meet demand at minimum cost. Demand for the services provided by these systems typically varies during the course of an operating day. The problem may involve 24 h continuous (cyclic) or less-than-24 h operations. Since its introduction by Edie,[2] both heuristic and integer programming based approaches have been studied for this problem. The integer programming based approaches have traditionally been based on the following set covering formulation provided by Dantzig.[1]

$$\min \sum_{k \in K} c_k X_k \tag{1}$$

subject to

$$\sum_{k \in K} a_{kt} X_k \geq b_t, \qquad \text{for all } t \in T \tag{2}$$

$$X_k \geq 0 \text{ and integer}, k \in K,$$

where K is the set of all shifts, T is the set of planning periods that the shift schedule covers, b_t is the number of employees needed in period t, c_k is the cost of assigning an employee to shift k, a_{kt} is one if period t is a work period for shift k and zero otherwise, and X_k is an integer decision variable representing the number of employees assigned to shift k.

Consideration of break flexibility has been shown to be important in determining shift schedules and minimum staff size to meet demand.[3,4] The case involving multiple relief and lunch breaks and break windows better matches scheduling practice. The set covering model, however, requires that all possible combinations of different shift types, lengths, shift start times as well as all possible relief and lunch break start time combinations be enumerated and included as different shifts in K. When break windows (time intervals during which employees should start and complete their breaks) and the resulting break placement possibilities are included, the number of shifts in K increases exponentially. Since the set covering formulation requires an integer variable for every shift in K, the size of the integer model makes obtaining an optimal shift schedule impractical in all but some small cases. Showalter and Mabert,[5] for example, recently stated that '*...Indiana Bell currently schedules staff where thousands of different shift schedules are present, due to 15 min planning periods, rest breaks, split shifts, and lunches. Since integer optimising procedures (i.e. branch and bound) requires significant computational effort to arrive at a solution when thousands of decision variables are present, we expect well-designated heuristics to be attractive for staff planners for some time to come*' (p 55). Similarly, Nanda and Browne[6] concluded that '*...Thus, while it is theoretically possible to use ILP (integer linear programming) for scheduling problems with variable breaks, for many applications it is*

Correspondence: Dr T Aykin, Engineering Research Center, Bell Laboratories, Princeton, NJ 08542 USA.

impractical to formulate, no less solve, by ILP' (p 206). Similar comments can be found in Bedworth and Bailey,[7] Glover *et al*,[8] Easton and Rossin[9] among others. This view has also been shared by a number of researchers who have recently developed and implemented industrial applications (see, for example, Holloran and Byrn,[10] p 13). We are also aware of two decision support systems recently developed for shift scheduling: NAMES™ 10 by AT&T for use at telemarketing centres to schedule agents (Spencer *et al*,[11]) and ROMAN by the Information Technology Institute, Singapore (Khoong and Lau[12]). These systems also use heuristics for scheduling shifts and breaks.

One popular strategy considered in the literature involves obtaining shift assignments by first solving a much smaller problem without including some or all relief and lunch breaks and break windows. A feasible shift schedule is then constructed by placing the relief and lunch breaks and adjusting the employee-shift assignments heuristically. Segal[13] applied this approach to a telephone operator scheduling problem in which each shift may include up to two 15 min relief breaks, one 30 min lunch break, and a 1.5 h break window for every relief/lunch break. He proposed a network flow based heuristic for solving the shift scheduling and the break placement problems separately in three phases. Keith[14] considered a similar operator scheduling problem with lunch and relief breaks at Illinois Bell Co. Assuming predetermined relief and lunch break start times (that is, ignoring break windows), the heuristic approach he proposed first solves the LP relaxation of a smaller set covering model and applies a rounding heuristic to obtain a feasible shift schedule. In the second part of his algorithm, break windows and alternative break placements are examined heuristically in an attempt to improve the shift schedule found in the first part. A number of other strategies have also been considered in the literature; Henderson and Berry,[15] and Bartoldi[16] considered heuristic solution of the complete set covering model. A construction type heuristic using the so-called Luce index was proposed by Buffa[17] for operator shift scheduling at the General Telephone Company of California.

Although the solution side of the problem has been studied extensively, the modelling side has not received similar attention. The integer programming based approaches have traditionally used the set covering model of Dantzig.[1] Modelling a special case of the problem with a single break has been considered by several researchers. Bailey and Fields[18] considered a shift scheduling problem involving shifts with a single break and a break window. They used the set covering approach and enumerated all break placement possibilities. This case was also considered by Bechtold and Jacobs.[19] They presented an integer programming formulation using a set of break placement variables. The formulation presented in Bechtold and Jacobs[19] was, however, limited to the cases involving less-than-24 h operations, one break of *identical* duration for all shifts, a single break window per shift, and no 'extraordinary break-window overlaps' (described as a situation in which the break window for a shift is included entirely in the break window of another shift and the larger break window starts and ends, respectively, at least one planning period earlier and later than the smaller break window). Recently, Thompson[20] combined the formulation of Bechtold and Jacobs[19] with that of Moondra[21] in a shift scheduling environment requiring a high degree of flexibility. Besides having the limitations of Bechtold and Jacobs,[19] the formulation presented by Thompson[20] is also limited by the fact that the degree of shift length and start time flexibility needed to realize the benefits of this approach is rare in practice. Moreover, due to the modelling approach used, the length of the break window has to decrease with the length of the shift (for example if a 9 h shift has a 3 h long lunch break window, a 7 h shift of the same type has to have a 1 h lunch break window). Another important limitation of this formulation (also pointed out in Thompson[20]) is its ability to model a realistic cost structure. In summary, past attempts to develop efficient modelling tools took advantage of the special structure of the cases considered and are not readily generalizable.

In the typical shift scheduling environment found in service industry, an employee working 7–9 consecutive hours a day receives one lunch break and two rest breaks, one before and one after the lunch. Shifts with shorter work span may be given fewer breaks. The length of a lunch break is usually a half hour to an hour and a rest break 15–30 min. Furthermore, when multiple breaks are given, usually separate break windows are specified for these breaks to provide scheduling flexibility and timely relief to employees.[6,7,13,15] Also, in some systems, namely telephone operator centres, employees may be assigned to split shifts consisting of two work periods separated by an idle period lasting 2–4 h. A rest break is also provided in each work period. In this paper, we propose a new approach for solving this problem (referred to as the general shift scheduling problem) optimally. Since, when formulated as a set covering model, the problem size presents the biggest obstacle to obtaining an optimal shift schedule, we first discuss an implicit formulation for the general shift scheduling problem (a discussion of the equivalence of these models together with an illustrative example is given in Aykin[3]). Rather than enumerating all possible break placements and associated shift combinations, two sets of variables, shift assignment variables and break variables, are used to assign employees to shifts and to schedule sufficient numbers of breaks for them. Compared to the set covering formulation of Dantzig,[1] this approach reduces the problem size greatly (see Tables 1 and 3). Furthermore, all shift scheduling problems that can be formulated with the set covering approach can be analyzed with this approach. The proposed branch-and-cut algorithm first solves the LP relaxation of the implicit formulation to obtain a lower

bound. An efficient rounding heuristic is then applied to the LP solution in an attempt to obtain a good upper bound and a feasible schedule. If this results in a solution with an objective value equal to the lower bound, the algorithm terminates with an optimal solution. Otherwise, it adds objective value cuts and upper bounds on the variables, and iteratively updates these cuts and bounds while using the rounding heuristic and a limited branch and bound search to find an optimal schedule. As reported in the computational results section, our experience with the branch and cut algorithm showed that the proposed approach is very efficient in solving large shift scheduling problems optimally. The scheduling problems we considered involve between 2160 and 32 928 shift variations. Some of these problems are, to the best of our knowledge, the largest shift scheduling problems solved optimally in the literature.

Problem formulation

To present the implicit formulation, we assume the following scheduling environment: Each employee receives two relief breaks and one lunch break; a work day is 24 h or less, divided into equal planning periods (eg 15 min intervals); each relief break covers one and each lunch break covers two planning periods. The formulation presented, however, is not restricted with these assumptions. Modelling shifts with more (or less) breaks and/or longer break durations is discussed in Aykin.[3] The following sets and parameters are needed to specify each shift $k \in K$:

c_k = the cost of assigning an employee to shift k
$a_{kt} = 1$ if period t is in the shift span of shift k (that is, a work or a break period) and zero otherwise
$B1_k$ = the sets of planning periods in which a worker assigned to shift k may start his/her first relief break
$B2_k$ = the sets of planning periods in which a worker assigned to shift k may start his/her second relief break
BL_k = the sets of planning periods in which a worker assigned to shift k may start his/her lunch break

Also define the following sets:

K = the set of all shift type-shift start time combinations
$T1_t$ = the set of shifts for which period t is a first relief break start period
$T2_t$ = the set of shifts for which period t is a second relief break start period
TL_t = the set of shifts for which period t is a lunch break start period

Finally, define the following decision variables:

X_k = the number of employees assigned to shift k
U_{kt} = the number of employees assigned to shift k and starting their first relief break in period t
V_{kt} = the number of employees assigned to shift k and starting their second relief break in period t
W_{kt} = the number of employees assigned to shift k and starting their lunch break in period t.

The break variables U_{kt1}, V_{kt2} and W_{kt3} are defined for $k \in K$ and for $t1 \in B1_k$, $t2 \in B2_k$ and $t3 \in BL_k$, since breaks can only be scheduled within their windows. The notation is illustrated in Figure 1.

Different than the set covering approach, the shift set K in the implicit formulation does *not* include the shift variations due to different break placements. Suppose there are q different shift types, and each shift type can start at s_h different times during a work day. Then, each shift type and shift start time for that type is identified as a different shift in K (namely K includes $\sum_{h=1}^{q} s_h$ shifts). This problem with multiple breaks and break windows can now be formulated as follows.

Problem P1

$$\min \sum_{k \in K} c_k X_k \tag{3}$$

subject to

$$\sum_{k \in K} a_{kt} X_k - \sum_{k \in T1_t} U_{kt} - \sum_{k \in T1_{(t-1)}} W_{k(t-1)} - \sum_{k \in TL_t} W_{kt} - \sum_{k \in T2_t} V_{kt} \geq b_t, \qquad \text{for all } t \in T, \tag{4}$$

$$X_k - \sum_{t \in B1_k} U_{kt} = 0, \qquad \text{for all } k \in K, \tag{5}$$

$$X_k - \sum_{t \in BL_k} W_{kt} = 0, \qquad \text{for all } k \in K, \tag{6}$$

$$X_k - \sum_{t \in B2_k} V_{kt} = 0, \qquad \text{for all } k \in K, \tag{7}$$

$X_k, U_{kt}, W_{kt}, V_{kt} \geq 0$ and integer for all k and t.

In Problem P1, constraint (4) ensures that the number of employees assigned to various shifts and available in period t minus the number of employees taking their first relief, lunch and second relief breaks is sufficient to meet the demand b_t in that period. On the left side of (4), the third and the fourth terms give the numbers of employees on their lunch break in period t including those who have started one period earlier in $(t-1)$ and continuing in period t, and those just starting in period t, respectively. Break constraints (5–7) require that one first relief, one lunch, and one second relief break be scheduled for every employee assigned to shift $k \in K$. The A-matrix (the matrix of constraint coefficients) for the two-shift example shown in Figure 1 is given in Figure 2.

Note that in the proposed formulation, although sufficient numbers of relief and lunch breaks are scheduled for all employees by the break constrains (5–7), their timing for an employee working a specific shift is not determined. In fact, provided constraints (5–7) are satisfied, assignment

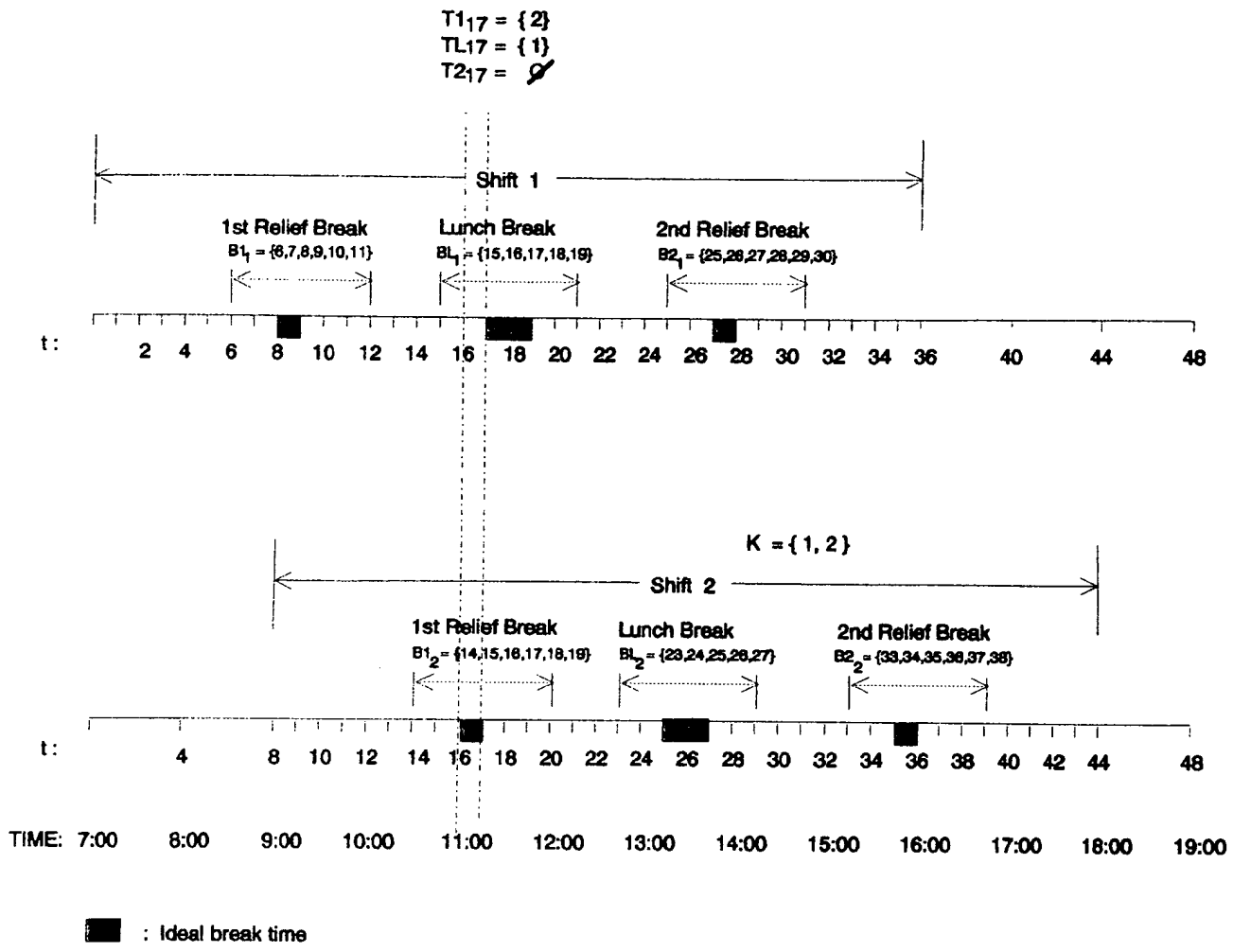


Figure 1 Shifts, break windows, break start times and the sets $T1_k$, TL_k, and $T2_k$.

of employees to specific break start time combinations does not affect the total number of employees needed to meet the demand during a work day. After obtaining an optimal solution to Problem P1, employees assigned to a shift can be assigned to specific break combinations in a variety of ways. One approach may involve assigning employees to the earliest first relief, lunch and second relief break times scheduled in the optimal solution for their shift. Since sufficient numbers of relief and lunch breaks are scheduled in the optimal solution for all shifts, this approach provides an efficient means for developing a complete schedule from an optimal solution of Problem P1.

Another version of the implicit formulation with fewer variables than both Problem P1 and the set covering formulation, and fewer constraints that Problem P1 can be obtained by using one of break constraints (5–7) to replace X_k with a set of break variables. This, however, increases the number of nonzeros and the density of the A-matrix. We didn't consider this version any further since our preliminary runs showed that the resulting formulation requires significantly more memory and time to solve than Problem P1.

Branch and cut algorithm

The branch and cut algorithm we propose first solves the LP relaxation of Problem P1. A lower bound is provided by this solution. Before introducing cuts and using a branch and bound procedure, an efficient rounding heuristic is applied to the LP solution to obtain a good upper bound and a feasible schedule. If the upper bound found by the heuristic equals the lower bound, the branch and cut algorithm terminates with an optimal solution. Otherwise, the schedule found by the heuristic is recorded and the objective value for this solution is stored as the upper bound. The branch and cut algorithm then adds cuts, updates the upper bounds on the variables, and solves its LP relaxation again. The rounding heuristic is then applied to this solution and the new schedule is recorded if it produces a tighter upper bound. The search is terminated if the new upper bound is equal to the lower bound. Otherwise, a limited branch and bound search is conducted to determine if a feasible solution with an objective value equal to the current lower bound exists. If one is found, the algorithm terminates with an optimal solution. If no such

	t	X_1	U_{17}	U_{18}	U_{19}	U_{110}	U_{111}	U_{112}	W_{116}	W_{117}	W_{118}	W_{119}	W_{120}	V_{126}	V_{127}	V_{128}	V_{129}	V_{130}	V_{131}	X_2	U_{215}	U_{216}	U_{217}	U_{218}	U_{219}	U_{220}	W_{224}	W_{225}	W_{226}	W_{227}	W_{228}	V_{234}	V_{235}	V_{236}	V_{237}	V_{238}	V_{239}
	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	5	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	6	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	7	1	−1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	8	1	0	−1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
e	9	1	0	0	−1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m	10	1	0	0	0	−1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a	11	1	0	0	0	0	−1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	12	1	0	0	0	0	0	−1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
d	13	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	14	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	15	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	−1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
o	16	1	0	0	0	0	0	0	−1	0	0	0	0	0	0	0	0	0	0	1	0	−1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	17	1	0	0	0	0	0	0	−1	−1	0	0	0	0	0	0	0	0	0	1	1	0	−1	0	0	0	0	0	0	0	0	0	0	0	0	0
s	18	1	0	0	0	0	0	0	0	−1	−1	0	0	0	0	0	0	0	0	1	1	0	0	−1	0	0	0	0	0	0	0	0	0	0	0	0
t	19	1	0	0	0	0	0	0	0	0	−1	−1	0	0	0	0	0	0	0	1	0	0	0	0	−1	0	0	0	0	0	0	0	0	0	0	0
r	20	1	0	0	0	0	0	0	0	0	0	−1	−1	0	0	0	0	0	0	1	0	0	0	0	0	−1	0	0	0	0	0	0	0	0	0	0
a	21	1	0	0	0	0	0	0	0	0	0	0	−1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i	22	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	23	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t	24	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	−1	0	0	0	0	0	0	0	0	0	0
s	25	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	−1	−1	0	0	0	0	0	0	0	0
	26	1	0	0	0	0	0	0	0	0	0	0	0	−1	0	0	0	0	0	1	0	0	0	0	0	0	−1	−1	0	0	0	0	0	0	0	0
	27	1	0	0	0	0	0	0	0	0	0	0	0	0	−1	0	0	0	0	1	1	0	0	0	0	0	0	−1	−1	0	0	0	0	0	0	0
	28	1	0	0	0	0	0	0	0	0	0	0	0	0	0	−1	0	0	0	1	0	0	0	0	0	0	0	0	−1	−1	0	0	0	0	0	0
	29	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	−1	0	0	1	0	0	0	0	0	0	0	0	0	−1	0	0	0	0	0	0
	30	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	−1	0	1	0	0	0	0	0	0	0	0	0	0	−1	0	0	0	0	0
	31	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	−1	1	0	0	0	0	0	0	0	0	0	0	0	−1	0	0	0	0
	32	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	−1	0	0	0
	33	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	−1	0	0
	34	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	−1	0
	35	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	−1
	36	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	C	1	−1	−1	−1	−1	−1	−1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	o	1	0	0	0	0	0	0	−1	−1	−1	−1	−1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
e	n	1	0	0	0	0	0	0	0	0	0	0	0	−1	−1	−1	−1	−1	−1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a	s	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	−1	−1	−1	−1	−1	−1	0	0	0	0	0	0	0	0	0	0
k	t.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	−1	−1	−1	−1	−1	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	−1	−1	−1	−1	−1	−1

Figure 2 The constraint coefficients for the 2 shift example.

solution exists, the lower bound and cuts, are updated and these steps are repeated. In the following sections, we present the initial bounds used in the algorithm, the rounding heuristic, and procedures for updating the lower and upper bounds. In this section, we assume that all objective function coefficients c_k (shift assignment costs) in (3) are equal (for convenience, to one). The case involving unequal assignment costs is discussed in the Appendix.

Initial bounds

A lower bound on the number of employees needed is obtained from the demand constraints (4) as

$$LB_0 = \left\{ \frac{\sum_{t \in T} b_t}{SL_{\max}} \right\} \tag{8}$$

where $SL_{\max} = \max_{k \in K}\{SL_k - (\text{number of break periods taken by employees assigned to shift } k)\}$ and SL_k is the number of planning periods covered by shift k.

One can also obtain upper bounds on the decision variables using again the demand constraints. Since, $X_k \geqslant U_{kt}$, $X_k \geqslant W_{kt}$ and $X_k \geqslant V_{kt}$ by definition, any upper bound on X_k is also an upper bound on the associated break variables. First, notice that if the employees working shift k are not given any break and are to work SL_k periods, say periods $t = 1, \ldots, SL_k$ without losing generality, then $X_k \leqslant b_{\max}$ where $b_{\max} = \max\{b_1, \ldots, b_{SLk}\}$. Thus, if $X_k = b_{\max}$, then employees working shift k will be sufficient to meet demand during periods $t = 1, \ldots, SL_k$ even without the employees assigned to other shifts. But, since the employees are given relief and/or lunch breaks, more employees may be needed especially during the break windows in order for every employee to take his/her breaks and to meet demand at the same time. Consider a

one-period relief break with a break window allowing n_{1k} possible break start times in $B1_k$. If this is the only break the employees assigned to shift k are given, then an upper bound on X_k is given by

$$UB1_k = \begin{cases} b_{\max}, & \text{if } \sum_{t \in B1_k} (b_{\max} - b_t) \geq b_{\max}, \\ b_{\max} + \left\{ \left\lceil \dfrac{(b_{\max} - \sum_{t \in B1_k} (b_{\max} - b_t))}{(n_{1k} - 1)} \right\rceil + 1 \right\}, & \text{otherwise.} \end{cases} \quad (9)$$

where $[y]$ is the largest integer less than or equal to y.

If the employees are given a lunch break lasting two periods, an upper bound on X_k can be obtained as follows. Let the maximum number of lunch breaks that can be scheduled with $b_{\max}$ employees be $L_{\max}$. If this is the only break that the employees take, then an upper bound is given by

$$UBL_k = \begin{cases} b_{\max}, & \text{if } L_{\max} \geq b_{\max}, \\ b_{\max} + \left\{ \left\lceil \dfrac{(b_{\max} - L_{\max})}{([(n_{Lk} + 1)/2] - 1)} \right\rceil + 1 \right\}, & \text{otherwise.} \end{cases} \quad (10)$$

An upper bound UB_k for a shift variable X_k including two or more breaks can be obtained by taking the highest of these bounds. The branch and cut algorithm initially computes UB_k for all shift variables X_k and include them as simple upper bounds.

Rounding heuristic

The rounding heuristic used in connection with the branch and cut algorithm provides an efficient means for obtaining a feasible schedule from the solution of the LP relaxation of Problem P1. As reported in the computational results section, solutions obtained with this heuristic provided tight upper bounds and, in many cases, turned out to be optimal.

The heuristic consists of two phases: A construction phase and an improvement phase. To start the construction phase, first the LP relaxation of the problem in the current iteration of the branch and cut algorithm is solved. Let X_k^{LP}, U_{kt}^{LP}, W_{kt}^{LP} and V_{kt}^{LP} be the values of the shift and break variables in one such solution. The shift variables X_k are rounded to the nearest integer; that is, if the fractional part of X_k^{LP} is less than 0.5, set $X_k = [X_k^{LP}]$. And if the fractional part of X_k^{LP} is greater than or equal to 0.5, then set $X_k = \langle X_k^{LP} \rangle$, where $\langle a \rangle$ is a function returning the smallest integer greater than or equal to a. The break variables are all rounded *down*, namely set $U_{kt} = [U_{kt}^{LP}]$, $W_{kt} = [W_{kt}^{LP}]$ and $V_{kt} = [V_{kt}^{LP}]$. Note that since the shift variables and the break variables are rounded differently and separately, not enough breaks may be scheduled for the employees assigned to shift k. The opposite (that is, availability of more breaks than the scheduled employees), however, is not possible. Therefore, before proceeding any further, the rounding heuristic schedules additional relief and lunch breaks whenever sufficient numbers of breaks are not scheduled for a shift. To schedule new breaks, the rounding heuristic determines the number of employees that are available (excluding the ones on break) minus the number needed in each planning period. Let S_t be this difference for period t (if positive, referred to as surplus, if negative, as shortage). Then, an additional break of the type needed for a shift is scheduled in the period with the highest surplus (smallest shortage) in the break window of that break type. And if two or more additional breaks are needed for a shift, they are scheduled in this fashion, one at a time, while updating S_k values. This procedure guarantees sufficient numbers of breaks for all scheduled employees while minimizing, one break at a time, the effects of scheduling additional breaks in terms of the number of employees needed.

After scheduling sufficient numbers of breaks for all shifts, the minimum period surplus/shortage $S_{\min} = \min_{t \in T}\{S_t\}$ is determined. If $S_{\min} < 0$, then the partial schedule available does not satisfy all period demand constraints (4). In this case a new employee is assigned to one of the shifts in an attempt to satisfy demand in all planning periods. To assign the new employee to a shift, first the minimum period surplus/shortage within the shift span of each shift, $S_k^{\min} = \min_{t \in T}\{a_{kt} S_t\}$, is determined. The new employee is then assigned to the shift with the largest period shortage, namely $S_{m^*} = \min_{k \in K} s_k^{\min}$. Ties are resolved using the total shortage covered by the shifts during their shift span. The number of employees assigned to shift m^* is increased by one, $X_{m^*} = X_{m^*} + 1$, and the rest and lunch breaks for the new employee are scheduled in the same way the additional breaks are scheduled after the initial rounding. We have considered a number of schemes for choosing shifts to assign new employees such as the total shortage covered during the shift span. Our preliminary tests, however, showed no improvement over the maximum period shortage approach discussed here.

After adding a new employee, the rounding heuristic checks for the redundant employees. This is done by considering dropping an employee from each shift (one shift at a time) while determining this employee's breaks (also to be dropped) as the ones that are scheduled in the periods with the smallest surplus (or highest shortage). When it is found that an employee can be removed from the current partial schedule without causing any additional shortage during his/her shift, the value of the associated shift variable X_k is decreased by one. The shift schedule is checked for redundant employees every time a new employee is added or when a feasible schedule satisfying

all period demand is found. When the latter condition occurs, the construction phase is completed.

In the improvement phase, the rounding heuristic attempts to lower the number of employees in the feasible schedule found in the construction phase by replacing two employees that are assigned to two different shifts by one new employee. For this purpose, two shifts with nonzero assignments in the current schedule are selected and one employee from each of these shifts is dropped together with the scheduled breaks for these employees. The new employee is added by considering all possible shifts. And the breaks for the new employee are scheduled to the periods in the respective break windows with the highest surplus (or smallest shortage if no period with surplus exists within the break window). If, after assigning the new employee to a shift, no period with shortage exists, the new schedule is feasible and replaces the current schedule with one less employee. The heuristic is terminated when no two employees can be replaced by one while maintaining feasibility. A flow chart of the rounding heuristic is given in Figure 3.

Updating lower and upper bounds

The lower bound on the objective value and the upper bounds on the decision variables are updated if the rounding heuristic can not locate an optimal solution in an iteration. Let Z_{LP} be the objective value for the LP solution found in the current iteration. Then the total number of employees needed is at least $LB = \langle Z_{LP} \rangle$. Hence, the proposed algorithm adds the following constraint to Problem P1

$$\sum_{k \in K} X_k = LB \tag{11}$$

Note that since the numbers of relief and lunch breaks that need to be scheduled should be equal to the number of

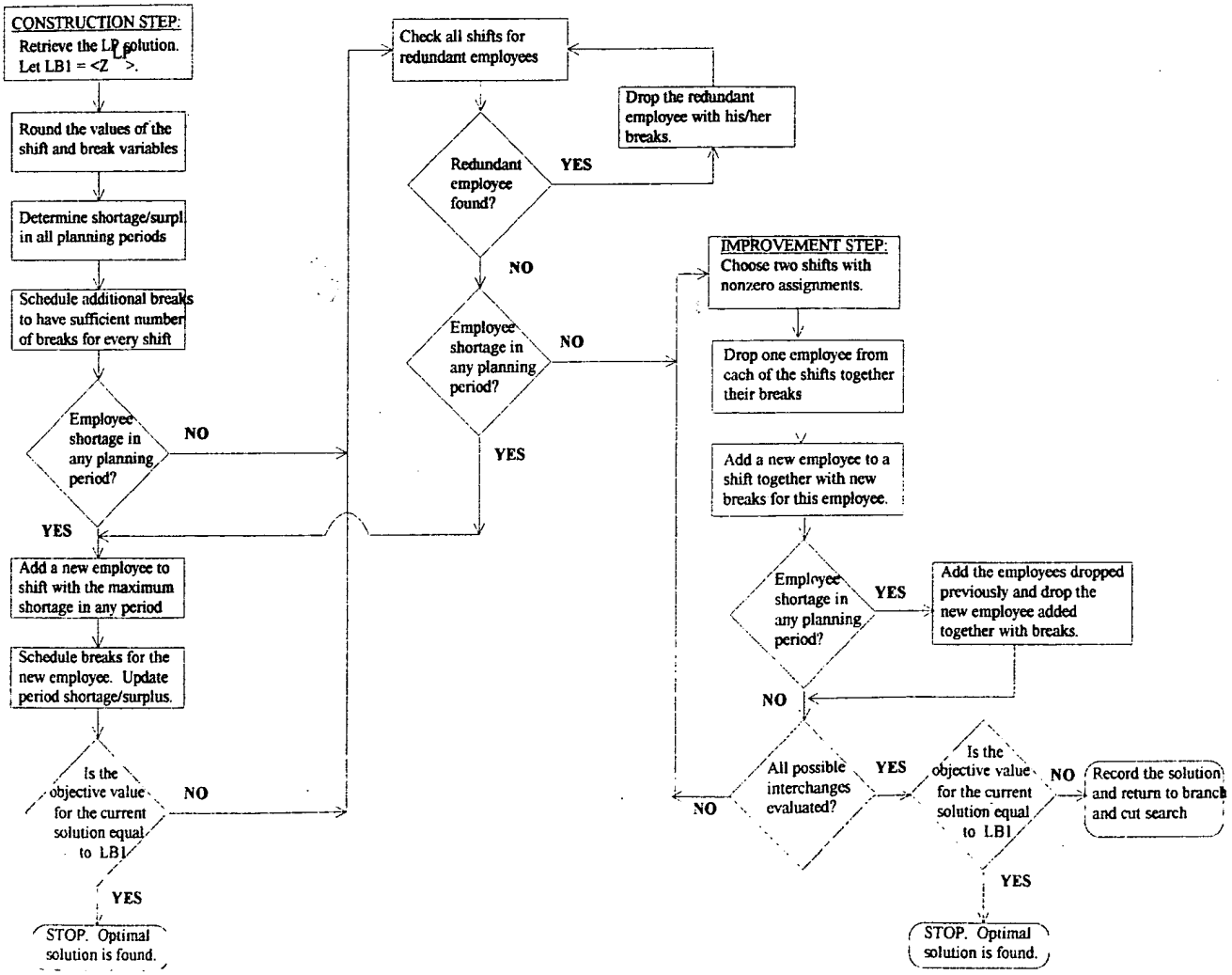


Figure 3 Flowchart for the rounding heuristic used in the branch and cut algorithm.

employees, the following constraints should also be satisfied

$$\sum_{k\in K}\sum_{j\in B1_k} U_{kj} = \mathrm{LB} \tag{12}$$

$$\sum_{k\in K}\sum_{j\in B2_k} V_{kj} = \mathrm{LB} \tag{13}$$

$$\sum_{k\in K}\sum_{j\in BL_k} W_{kj} = \mathrm{LB} \tag{14}$$

Using the objective value cut (11), upper bounds on the shift variables X_k can be updated as follows. Suppose shift k covers SL_k planning periods (without losing generality, $t = 1, \ldots, SL_k$). Now consider the demand constraints (4) for the planning periods *not* covered by shift k; that is, the demand constraints for the periods $t = (SL_k + 1), \ldots, n_T$, where n_T is the number of periods in T. By aggregating these constraints, we obtain

$$\sum_{t=(SL_k+1)}^{n_T}\sum_{j\in K:\, j\neq k} a_{jt}X_j - \sum_{t=(SL_k+1)}^{n_T}\sum_{i\in T1_t} U_{it} - \sum_{t=(SL_k+1)}^{n_T}\left(\sum_{i\in TL_{(t-1)}} W_{i(t-1)} + \sum_{i\in TL_t} W_{it}\right) - \sum_{t=(SL_k+1)}^{n_T}\sum_{i\in T2_t} V_{it} \geqslant \sum_{t=(SL_k+1)}^{n_T} b_t \tag{15}$$

Note that if another shift, say j, and shift k have no common planning periods then the variables associated with shift j on the left side of (15) can be regrouped as

$$SL_jX_j - \sum_{t\in B1_j} U_{jt} - 2\sum_{t\in BL_j} W_{jt} - \sum_{t\in B2_j} V_{jt} \tag{16}$$

Using (5 to 7) and (16) can be simplified as

$$(SL_j - 4)X_j \tag{17}$$

where $(SL_j - 4)$ is the number of work periods for shift j, with four periods (out of SL_j) reserved for relief and lunch breaks (for shifts with longer or more breaks, this is replaced by the total break duration for the shift). If, on the other hand, shifts j and k have some common planning periods, say Δ periods, then the variables associated with this shift in (15) can be regrouped as

$$(SL_j - \Delta)X_j - \sum_{t\in B1'_j} U_{jt} - \sum_{t\in B1'_j} W_{jt} - \sum_{t\in B2'_j} V_{jt} \tag{18}$$

where $B1'_j$, BL'_j and $B2'_j$ are the sets of planning periods in the break windows for the first relief, lunch and second relief breaks of shift j that are not in the shift span of k. Note that (18) is always less than or equal to $\{(SL_j - 4)X_j\}$ since this is the maximum number of planning periods the employees assigned to shift j are available for work. As in (18), if one considers a particular part of shift j during a work day, the number of work periods in that part will be less than this quantity. Now using this together with (17), (15) can be rewritten as

$$\sum_{j\in K:\, j\neq k}(SL_j - 4)X_j \geqslant \sum_{t=(SL_k+1)}^{n_T} b_t \tag{19}$$

Let $SL_{\max} = \max_{j\in K\neq k}\{SL_j - 4\}$. Then, (19) can be written as

$$\sum_{j\in K:\, j\neq k} X_j \geqslant \frac{\sum_{t=(SL_k+1)}^{n_T} b_t}{SL_{\max}} \tag{20}$$

Since $\sum_{k\in K} X_k = LB$ by (11), a new upper bound on X_k is obtained using (20) as follows

$$X_k \leqslant \mathrm{LB} - \frac{\sum_{t\in D_k} b_t}{SL_{\max}} \tag{21}$$

where D_k is the set of periods not covered by shift k. Our experience with the problems solved in this study showed that (21) provides a tighter upper bound for X_k than the initial upper bound obtained from (9–10). Occasionally, however (21) may not produce a tighter bound (for example, as LB is increased in the subsequent iterations of the branch and cut algorithm). In that case, the lower of these two bounds is taken as the upper bound.

Branch and cut algorithm

We now outline the proposed branch and cut algorithm for Problem P1.

Step 0. Calculate and include the initial upper bounds UB_k using (9) and (10) for X_k in Problem P1.

Step 1. Solve the LP relaxation of Problem P1. Let the objective value for the optimal LP solution be Z_{LP}. If the LP solution is integer, stop; an optimal solution is found. Otherwise, let $LB = \langle Z_{LP}\rangle$ and apply the rounding heuristic. Let the objective value for the heuristic solution be OBJVAL. If OBJVAL = LB, stop; an optimal solution is found. Otherwise, record the solution and go to Step 2.

Step 2. Add constraints (11–14) and update the upper bounds for the variables using LB and (21).

Step 3. Solve the LP relaxation of P1 with the new cuts. If the LP solution is integer, stop; an optimal solution is found. Otherwise, apply the rounding heuristic to the new LP solution. If a solution with an objective value lower than OBJVAL is found, update the solution and OBJVAL. If OBJVAL = LB, stop; an optimal solution is found. Otherwise to go Step 4.

Step 4. Apply branch and bound to determine if a feasible solution with an objective value of LB exists. If a solution with an objective value of LB is located, stop; an optimal solution is found. Otherwise, go to Step 5.

Step 5. Set LB = LB + 1, and update the right hand side of the objective value cuts added in Step 2. Update

the upper bounds for the variables using LB and (21), and go to Step 3.

The branch and cut algorithm stops when the LP solution is integer, a heuristic solution with OBJVAL equal to LB is found, or a feasible solution with an objective value of LB is found in Step 4. In all these three cases, the solution has an objective value equal to the lower bound and, therefore, is an optimal solution to Problem P1.

Computational results

We studied the performance of the proposed branch and cut algorithm by solving two groups of problems containing a total of 90 cases. The branch and cut algorithm was coded in FORTRAN. This program generates and inputs problem data, calls various LINDO subroutines for solving linear programming relaxation and branch and bound search, and applies the rounding heuristic. All test problems were solved in batch mode on a Sun Sparc 512 computer.

Scheduling environment

The scheduling environment chosen for this study involves a continuous 24 h work day (cyclical scheduling environment) to test the proposed approach in large scheduling problems. Employee requirements are determined for 15 min planning periods ($4 \times 24\,\text{h} = 96$ planning periods) indexed successively $T = \{1, \ldots, 96\}$. Concerning the shift start times, we considered two groups of test problems while keeping all the other aspects of the scheduling environment the same: In the first group of problems, shifts are assumed to start on the hour and on the half hour, resulting in 48 shift start times in a work day. In order to test the proposed approach in even larger cases, in the second group of problems, shifts are assumed to start every 15 minutes, resulting in 96 shift start times in a work day. Duration of a shift is taken as 9 h including breaks. It is assumed that each employee is given one half-hour lunch break (two 15 min periods) and two 15 min relief breaks, one before the lunch break and one after. We specify the break windows for each shift with reference to ideal break times as follows: the ideal start time for the first relief break is specified as 2 h after the start of a shift, the ideal lunch break time is located in the middle of the work day after 4 h of work (that is, after 4 h plus 15 min for the first relief break) and the ideal second relief break time is specified as 2 h after the completion of the lunch break starting at the ideal lunch time. The break windows are formed by setting the earliest break start times half an hour earlier than the ideal break times. The lengths of break windows are varied to carry out the computational experiments with problems of varying difficulty level. The relief and lunch break windows considered allow 3–7 break start times (see Tables 1 and 3). We considered five different demand patterns, resulting in 60 problems in the first problem group and 30 problems in the second group. Three of the demand patterns considered were found in the literature and were either obtained directly from or resembling to real patterns from service systems: (i) the trimodal demand profile for telephone operators provided by Segal[13] with a minimum quarter hour demand of 2 and a maximum of 89 employees, (ii) the bimodal demand profile with a minimum quarter hour demand of 2 and a maximum of 9 obtained from the General Telephone and Electronics Company of California and given in Henderson and Berry,[15] and (iii) the unimodal demand profile given by Buffa[17] with a minimum quarter-hour demand of 2 and a maximum of 27 that was obtained from the General Telephone Company of California. In addition to these three demand patterns, we considered two synthetically generated patterns; a bimodal and a trimodal sinusoidal demand patterns with an average quarter-hour demand of 27.5. Minimum and maximum quarter-hour demand across these two demand patterns were 5 and 50, respectively. Like in Henderson and Berry,[15] the cost of assigning employees to shifts is assumed to be the same and equal to one.

Computational results with the test problems

The number of variables, number of constraints, number of nonzero coefficients in the A-matrix and its density for Problem P1 and the set covering model are shown in Table 1 for the first group of problems and in Table 3 for the second group of problems. As seen in Tables 1 and 3, the implicit formulation requires substantially smaller, between 73.3–93.6% less, number of variables than the equivalent set covering formulation. The problems characteristics shown in Tables 1 and 3 for the set covering approach exceed the limits of most hardware/software configurations including the one used in this study. Hence, solving these test problems optimally with the set covering approach was not possible.

The minimum, average and maximum solution times for the first group of problems are shown in Table 2. Also reported in this table are the minimum, average and maximum LP solution times and the heuristic search times within the branch and cut algorithm. Our preliminary test runs showed that break variables U_{kt}, W_{kt} and V_{kt} tend to be integer when the associated shift variable X_k has an integer value. Therefore, a branching strategy giving higher priority to shift variables X_k was implemented in the branch and bound search with LINDO (step 4 of the proposed algorithm). When necessary, we made two attempts to solve a problem with two different branching strategies. In the first attempt, the shift variables were selected for branching according to their start times. We set the CPU time limit to 500 s. When the proposed branch and cut algorithm could not locate an optimal solution in the first attempt within this time limit, a second attempt was made by selecting shift

Table 1 Set covering *vs* proposed model: problem characteristics for the first group of problems

		Number of variables			*Number of constraints*		*Number of nonzeros in A-matrix*			*% nonzeros in A-matrix*	
$n_{1k} = n_{2k}$	n_{Lk}	*Set covering*	*Proposed model*	*% reduction*	*Set covering*	*Proposed model*	*Set covering*	*Proposed model*	*% reduction*	*Set covering*	*Proposed model*
3	5	2160	576	73.3	96	240	60120	3168	95.4	33.3	2.3
4	4	3072	624	79.7	96	240	98304	3216	96.7	33.3	2.1
4	5	3840	672	82.5	96	240	122880	3360	97.3	33.3	2.1
4	6	4608	720	84.4	96	240	147456	3504	97.6	33.3	2.0
5	3	3600	672	81.4	96	240	115200	3264	97.2	33.3	2.0
5	4	4800	720	85.0	96	240	153600	3408	97.8	33.3	2.0
5	5	6000	768	87.2	96	240	192000	3552	98.2	33.3	1.9
5	6	7200	816	88.7	96	240	230400	3696	98.4	33.3	1.9
6	5	8640	864	90.0	96	240	276480	3744	98.6	33.3	1.8
6	6	10368	912	91.2	96	240	331776	3888	98.8	33.3	1.8
7	6	12096	1008	91.7	96	240	387072	4080	98.9	33.3	1.7
7	7	16464	1056	93.6	96	240	526848	4224	99.2	33.3	1.7

variables for branching according to the total demand covered by the associated shifts. When there is a tie, the shift variable with the earliest shift start time is given higher priority. If a second attempt was needed, the solution time reported for that problem is the total time for the two attempts. Hence, the CPU time allowed for two attempts was 1000 s.

With the branch and cut algorithm, out of 60 problems considered in the first group, 54 were solved optimally in the first attempt and 4 more in the second attempt. In only 2 problems out of 60, one involving $n_{ik} = n_{2k} = 6$ and $n_{Lk} = 5$ and the other $n_{ik} = n_{2k} = 5$ and $n_{Lk} = 3$, an optimal solution was not located after two attempts within the time limit. In these two cases, however, the best feasible schedule found by the branch and cut algorithm (with the rounding heuristic) required only one more employee than the best lower bound. Thus, in these two cases, the best solution obtained with the branch and cut algorithm was the closest possible solution ('best nonoptimal' solution) to the optimal solution in terms of the number of employees needed.

The rounding heuristic used in the branch and cut algorithm is useful for a number of reasons. First, it is very efficient and quite often locates an optimal solution; it located an optimal solution in 28 (out of 60) of the problems considered in this group. When this is not the case, this heuristic provides very good feasible solutions and tight upper bounds. In fact in all but one of the remaining 32 cases, the best heuristic solution required only one more employee than the best lower bound. Thus, in the worst case, the heuristic solutions obtained in these cases were the best nonoptimal solutions possible.

We analysed the performance of the branch and cut algorithm further by plotting a histogram of the total solution times (Figure 4). As seen from this histogram, in 70% of the problems (that is, 42 out of 60 problems), an optimal solution was obtained within 40 CPU s, and in

Table 2 LP solution, heuristic search and total solution times with the branch and cut algorithm (CPU s)—first group of problems

Break Window Combination	$n_{1k} = n_{2k}$	n_{Lk}	*LP solution time*			*Heuristic search time*			*Total solution time with the branch and cut algorithm*		
			Min	*Average*	*Max*	*Min*	*Average*	*Max*	*Min*	*Average*	*Max*
1	3	5	2.90	6.42	12.96	*	0.28	0.98	3.23	14.60	47.61
2	4	4	2.76	8.02	22.95	*	0.72	2.05	2.90	119.33	522.31
3	4	5	3.51	8.41	18.30	1.07	1.66	2.49	17.44	45.89	99.94
4	4	6	2.96	4.85	7.49	0.64	1.69	2.38	14.71	84.90	185.29
5	5	3	2.90	10.76	25.62	0.36	1.41	2.03	21.08	299.09	747.72
6	5	4	3.27	7.38	18.61	0.77	1.35	2.59	10.45	136.54	497.04
7	5	5	3.94	9.28	15.66	1.08	1.62	2.73	13.84	30.17	37.89
8	5	6	2.31	4.30	9.05	0.02	1.23	2.30	2.53	116.15	504.08
9	6	5	5.39	10.21	16.75	0.04	1.32	2.67	7.24	18.19	29.35
10	6	6	3.59	5.16	7.92	0.04	0.94	2.98	3.59	33.78	130.40
11	7	6	2.77	5.95	13.49	0.04	0.61	1.90	3.92	137.11	439.30
12	7	7	1.97	5.35	7.72	0.04	0.45	1.02	2.01	117.93	571.80

* <0.01 s.

Table 3 Set covering *vs* proposed model: problem characteristics for the large problems

		Number of variables			*Number of constraints*		*Number of nonzeros in A-matrix*			*% nonzeros in A-matrix*	
$n_{1k} = n_{2k}$	n_{Lk}	*Set covering*	*Proposed model*	*% reduction*	*Set covering*	*Proposed model*	*Set covering*	*Proposed model*	*% reduction*	*Set covering*	*Proposed model*
5	5	12000	1536	87.2	96	384	384000	7104	98.2	33.3	1.9
5	6	14200	1632	88.7	96	384	460800	7392	98.4	33.3	1.9
6	5	17280	1728	90.0	96	384	552960	7488	98.6	33.3	1.8
6	6	20736	1824	91.2	96	384	663552	7776	98.8	33.3	1.8
7	6	24192	2016	91.7	96	384	774144	8160	98.9	33.3	1.7
7	7	32928	2112	93.6	96	384	1053696	8448	99.2	33.3	1.7

about 78% (47 out of 60), within 100 CPU s. In only about 22% (13 out of 60) of the problems considered, the time needed exceeded 100 CPU s.

In the second group, we considered 30 large shift scheduling problems involving 96 shift start times and the largest 6 break window combinations from Table 1. The branch and cut algorithm performed remarkably well in these large problems; 24 out of 30 problems were solved optimally in the first attempt, and 3 more in the second attempt. Only in three cases an optimal solution was not located. However, the best solution obtained with the rounding heuristic in these three cases, required again one more employee than the best lower bound. Thus, in the worst case, the 'best nonoptimal' solutions were obtained. Concerning the overall performance of the rounding heuristic, we observed that it located an optimal solution in 14 out of 30 problems. And in the remaining 16 cases, the rounding heuristic located the 'best nonoptimal' solutions. The minimum, average and maximum LP solution times, heuristic search times and the total solution times with the branch and cut algorithm for the second group of problems are shown in Table 4. For this group of problems, we set the CPU time limit to 750 CPU s. Note that the solution time does not seem to be increasing with the problem size, although the times reported are somewhat higher compared to the first group of problems (Table 2). This is expected since the size of the problems in the second group is substantially larger (containing as many as 32 928 shift variations) than those in the first group. The distribution of the total solution times was as follows; 11 problems (37%) required less than 40 CPU s and 18 cases (60%) less than 100 CPU s.

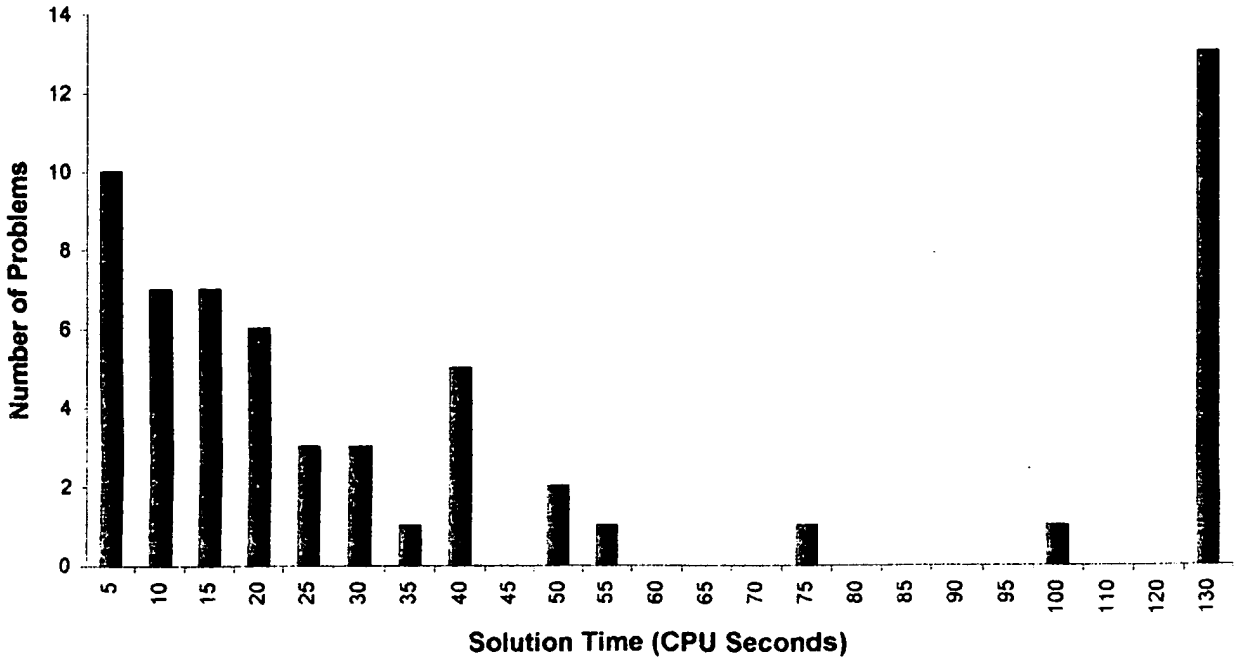


Figure 4 Distribution of the solution times—first group of problems.

Table 4 LP solution, heuristic search and total solution times with the branch and cut algorithm (CPU s)—large problems

		LP solution time			*Heuristic search time*			*Total solution time with the branch and cut algorithm*		
$n_{1k} = n_{2k}$	n_{Lk}	*Min*	*Average*	*Max*	*Min*	*Average*	*Max*	*Min*	*Average*	*Max*
5	5	9.29	26.08	68.93	0.02	5.96	10.30	25.02	235.91	871.26
5	6	7.04	31.40	65.32	0.02	5.12	10.64	7.09	118.31	372.00
6	5	5.83	23.15	38.55	0.05	3.95	7.48	5.97	93.13	253.58
6	6	7.59	32.91	62.28	0.02	3.50	9.43	7.61	380.22	1403.61
7	6	8.81	21.08	59.97	0.04	2.18	10.16	8.99	167.02	437.27
7	7	5.10	19.69	31.73	0.42	4.36	9.68	6.55	228.49	888.23

Conclusions

In this paper, we presented a new branch and cut algorithm for optimal shift scheduling. We tested the new approach by solving two groups of test problems involving a total of 90 cases. The problems considered involved as many as 32 928 shift variations. These problems, to the best of our knowledge, are the largest solved optimally in the literature. Our results showed that with the implicit formulation and the branch and cut algorithm, large shift scheduling problems can be solved optimally. Our results also showed that the rounding heuristic used in the branch and cut algorithm often locates an optimal solution frequently. And when this is not the case, it is most likely to locate the 'best nonoptimal' solution. The proposed approach may be extended in a number of directions. An important extension would involve its application to the tour scheduling problem. The tour scheduling problem involves both daily shift scheduling for the work days in a week and days-off scheduling. Another area in which more work needs to be done is the development of decision support systems offering various scheduling and modelling flexibilities. This would be particularly important for the service organizations such as telephone operator centres and telemarketing centres where the workforce may account up to 60% or more of total operating expenses.

Appendix

Let H be the set of shift types with different wage rates. There may be several shift start times associated with each shift type in H. Let set K again contain all shift type-shift start time combinations. We are assuming that all employees assigned to a particular shift type in H are paid the same wage $c_h, h \in H$. An initial lower bound on the objective value in this case is obtained as

$$\mathrm{LB}_0 = \left\{ \frac{c_{\min} \sum_{t \in T} b_t}{SL_{\max}} \right\} \tag{A1}$$

where $c_{\min} = \min_{h \in H}\{c_h\}$.

The initial upper bounds calculated using (9) and (10) are independent of c_h. Hence, they are valid in this case with unequal shift assignment costs. In Step 1 of the branch and cut algorithm, again the LP relaxation of P1 is solved and a lower bound from the optimal LP solution is obtained. But this time we set $\mathrm{LB} = Z_{\mathrm{LP}}$. If the solution is integer, then the algorithm terminates with the optimal shift schedule. Otherwise, the following integer program is solved in order to obtain a lower bound on the objective value in this case.

Problem P2

$$\min \sum_{h \in H} c_h Y_h \tag{A2}$$

subject to

$$\sum_{h \in H} c_h Y_h \geqslant \mathrm{LB}_1 + \delta \tag{A3}$$

$$Y_h \geqslant 0 \text{ and integer for } h \in H,$$

where Y_h is an integer variable defined as the number of employees assigned to shift type $h, h \in H$. In the first iteration of the branch and cut algorithm, δ is set to zero in Step 1. And in the subsequent iterations, $\delta = \min_{h1, h2 \in H}\{|c_{h1} - c_{h2}|\}$. The objective value of an *integer* optimal solution to P2 provides a lower bound LB on the value of objective function (3) since the demand constraints (4) and the break constraints (5–7) are not taken into consideration. Note that when all shift assignment costs are equal and set to $c_h = 1, h \in H$, solution of Problem P2 can be easily determined as described in Section 3. With the lower bound LB obtained from the optimal solution of Problem P2, the following objective value cuts are added to Problem P1.

$$\sum_{k \in K} c_k X_k = \mathrm{LB} \tag{A4}$$

$$\sum_{k \in K} \sum_{j \in Bl_k} c_k U_{kj} = \mathrm{LB} \tag{A5}$$

$$\sum_{k \in K} \sum_{j \in BL_k} c_k W_{kj} = \mathrm{LB} \tag{A6}$$

$$\sum_{k \in K} \sum_{j \in B2_k} c_k V_{kj} = \mathrm{LB} \tag{A7}$$

To update upper bounds for the variables X_k in this case, from (19) we obtain

$$c_{\min} \sum_{j \in K; j \neq k} (SL_j - 4)X_j \geqslant c_{\min} \sum_{t \in D_k} b_t \qquad \text{(A8)}$$

where $c_{\min}$ is as defined before. But since $c_j \geqslant c_{\min}$, $j \in K$,

$$\sum_{j \in K; j \neq k} c_j(SL_j - 4)X_j \geqslant c_{\min} \sum_{t \in D_k} b_t \qquad \text{(A9)}$$

And since $SL_{\max} \geqslant SL_j$, $\in K$, we obtain

$$\sum_{j \in K; j \neq k} c_j X_j \geqslant \left\{ \frac{c_{\min} \sum_{t \in D_k} b_t}{SL_{\max}} \right\} \qquad \text{(A10)}$$

Now combining (A10) with (A4), the following upper bound is obtained.

$$X_k \leqslant \frac{\text{LB} - c_{\min} \sum_{t \in D_k} b_t / SL_{\max}}{c_k}, \qquad \text{(A11)}$$

If an optimal solution to P1 is not found in an iteration of the branch and cut algorithm, the lower bound LB_1 is updated in Step 5 by solving Problem P2. The new lower bound is then used to update both constraints (A4–A7) and the upper bounds.

References

1 Dantzig GB (1954). A comment on Edie's 'Traffic delays at toll booths'. *Ops Res* **2**: 339–341.
2 Edie LC (1954). Traffic delays at toll booths. *Ops Res* **2**: 107–138.
3 Aykin T (1996). Optimal shift scheduling with multiple break windows. *Mgmt Sci* **42**: 591–603.
4 Thompson GM (1988). *A comparison of techniques for scheduling non-homogeneous employees in a service environment subject to non-cyclical demand, Unpublished Ph.D. dissertation, Florida State University.*
5 Showalter MJ and Mabert VA (1988). An evaluation of a full/part-time tour scheduling methodology. *Int J Ops & Prod Res* **8**: 54–71.
6 Nanda R and Browne J(1992). *Introduction to Employee Scheduling.* Van Nostrand Reinhold: New York, N.Y.
7 Bedworth DD and Bailey JE(1987). *Integrated Production Control Systems*, 2nd ed. John Wiley and Sons: New York.
8 Glover F, McMillan C and Grover R (1985). A heuristic programming approach to the employee scheduling problem and some thoughts on 'Managerial Robots'. *J Ops Mgmt* **4**: 113–128.
9 Easton FF and Rossin DF (1991). Sufficient working subsets for the tour scheduling problem. *Mgmt Sci* **37**: 1441–1451.
10 Holloran TJ and Byrn JE (1986). United Airlines station manpower planning. *Interfaces* **16**: 39–50.
11 Spencer T, Brigandi T, Barlow G and Gomez R (1992). A decision support system for workforce management. In *Modelling and Simulation.* Proceedings of the 23rd annual Pittsburg conference, part 5.
12 Khoong CM and Lau HC (1992). ROMAN: An integrated approach to manpower planning and scheduling. In: Balci O, Sharda R and Zenios S (eds). *Computer Science and Operations Research: New Developments in Their Interfaces.* Pergamon Press: New York.
13 Segal M (1974). The operator-scheduling problem: A network flow approach. *Ops Res* **22**: 808–823.
14 Keith EG (1979). Operator scheduling. *AIII Trans* **11**: 37–41.
15 Henderson WB and Berry WL (1976). Heuristic methods for telephone operator shift scheduling: An experimental analysis. *Mgmt Sci* **22**: 1372–1380.
16 Bartoldi JJ (1981). A guaranteed-accuracy round-off algorithm for cyclic scheduling and set covering. *Ops Res* **29**: 501–510.
17 Buffa ES (1980). *Modern Production and Operations Management*, 6th ed. John Wiley and Sons: New York.
18 Bailey J and Fields J (1985). Personnel scheduling with flexshift models. *J Ops Mgmt* **5**: 327–338.
19 Bechtold SE and Jacobs LW (1990). Implicit modeling of flexible break assignments in optimal shift sheduling. *Mgmt Sci* **36**: 1339–1351.
20 Thompson GM (1995). Improved implicit optimal modelling of labor shift scheduling problem. *Mgmt Sci* **41**: 595–607.
21 Moondra SL (1976). An L.P. model for work force scheduling for banks. *J Bank Res* **6**: 299–301.

Received April 1995;
accepted June 1997 after two revisions

Vol. 36, No. 11, November 1990
Printed in U.S.A.

IMPLICIT MODELING OF FLEXIBLE BREAK ASSIGNMENTS IN OPTIMAL SHIFT SCHEDULING*

STEPHEN E. BECHTOLD AND LARRY W. JACOBS
Florida State University, College of Business, Tallahassee, Florida 32306-1402
Northern Illinois University, College of Business, DeKalb, Illinois 60115

The labor scheduling literature has demonstrated that the use of flexibility in designing employee schedules can result in a substantial improvement in labor utilization. This paper presents a new implicit integer linear programming formulation for the inclusion of meal/rest-break flexibility. Although the use of flexible break assignments in labor staffing decisions has been of research interest since an early article by Segal (1974), due to problem size, the majority of related research has involved the use of heuristics. An experimental analysis using four different labor requirements patterns and ten shift-length combinations demonstrated that, when flexible break assignments were modeled, the implicit formulation was superior to the traditional set-covering formulation with respect to 1) execution time, 2) computer memory requirements, and 3) the ability to produce optimal integer solutions to larger problems incorporating greater flexibility. Finally, a number of possible extensions of the implicit modeling approach for use in other labor scheduling environments are identified.

(SERVICE OPERATIONS; LABOR SCHEDULING; INTEGER PROGRAMMING)

1. Introduction

Service organizations frequently consider a large number of alternative work shifts in the development of labor schedules. These alternatives are typically developed through the use of variations in starting times, finish times, and the placement of breaks. While the existence of a large number of shift alternatives increases the complexity of the scheduling task, it also offers an opportunity to improve labor utilization (Byrne and Potts 1973; Showalter and Mabert 1988).

This paper is focused upon the problem of obtaining optimal integer solutions in shift scheduling environments where a high degree of scheduling flexibility is desired. Specifically, we introduce a new implicit integer programming approach to modeling break-placement flexibility.

Integer programming approaches to the shift scheduling problem have traditionally been based upon the set-covering formulation originally proposed by Dantzig (1954). If break placement is not modeled, solutions can be efficiently obtained with a network flow approach (Segal 1974). However, Thompson (1988) has concluded that it is important, relative to the objective of minimizing the amount of labor scheduled, to model at least some subset of the allowable break options within the optimization model.

Unfortunately, the use of flexible break assignments can dramatically increase the number of integer variables required in the set-covering formulation. For example, in a telephone operator shift scheduling environment, over 98 percent of the 7,120 shift variables associated with allowed shifts were required as a direct result of modeling flexible break assignments (Henderson and Berry 1976). The set-covering formulation has generally been found to be intractable with respect to obtaining optimal integer solutions in such environments. Thus, given the expected benefits of scheduling flexibility, organizations have tended to prefer heuristic solutions to large problems over optimal solutions to small problems (Glover, McMillan, and Glover 1984; Glover and McMillan 1986; Holloran and Byrn 1986; Taylor and Huxley 1989).

* Accepted by Stephen C. Graves; received September 23, 1988. This paper has been with the authors 5 months for 1 revision.

0025-1909/90/3611/1339$01.25

A limited number of research papers have appeared in the literature which have directly addressed the size related problems associated with modeling of flexible break assignments. Segal (1974) developed a three-phased heuristic which used the solutions of network flow models as the basis for an iterative break assignment procedure. Moondra (1976) attempted to provide some degree of flexibility, without increasing the number of problem variables, by assuming that half of the employees in a given shift would take their break in one planning period with the remainder of the employees taking their break in the immediately following planning period. Keith (1979) developed linear programming relaxation heuristics where only one of the allowed break assignment combinations was initially included in the model. Adjustments in the actual break assignments were made during the heuristic phase of the procedure.

Gaballa and Pearce (1979) developed an integer programming formulation which modeled break-placement flexibility by including a separate break variable for every period for which a break is allowed for each shift. Although the model incorporated a unique concept, the associated number of integer variables was greater than the number which would have been required by an equivalent set-covering model.

The development of the new implicit modeling approach was based upon the realization that it was likely that substantial reductions in model size could only be achieved by reducing the information requirements of the model. A large reduction in the information requirements was achieved in the new modeling approach through implicit representation of break assignments for all shifts. Specifically, this was accomplished by associating break variables with planning periods as opposed to shifts. A particular break variable represents the total number of employees starting a break in its associated planning period. Thus, break assignments are not made until the optimal solution to the implicit model has been obtained.

The new implicit model eliminates the major disadvantage in Gaballa and Pearce (1979). Moreover, the definition of break variables in the new model as the number of breaks initiated in appropriate planning periods permits the modeling of breaks with durations greater than or equal to the length of the planning period modeled.

In the following section, we define critical terms, state our initial modeling assumptions, and present the new implicit model. §3 presents the results of two experimental analyses which were used to compare the implicit and set-covering approaches to shift scheduling. The first provides a comparison of problem sizes and execution times for the two approaches across a variety of operating conditions. The second demonstrates the robustness of the implicit modeling approach with respect to its capability to generate optimal solutions under conditions which violate a key assumption. §4 summarizes the experimental results and identifies a number of useful extensions of the implicit modeling approach.

2. Modeling

In this section, we develop a new model in which the assignment of a single break to each employee scheduled is implicit in lieu of its explicit representation in the traditional set-covering formulation. Since a single break is being modeled, it may be assumed to be a meal break.

For purposes of comparison, we display the Dantzig set-covering formulation for shift scheduling in §2.1. §§2.2 and 2.3 provide the necessary foundation for the new model which is developed in §2.4. The break assignment algorithm is described in §2.5.

2.1. *Set-Covering Formulation*

The Dantzig set-covering formulation (P1) for shift scheduling is typically expressed:

$$\text{minimize} \quad \sum_{j \in J} c_j y_j, \tag{1}$$

$$\text{s.t.} \quad \sum_{j \in \mathbf{J}} a_{ij} y_j \geq r_i \quad \text{for all} \quad i \in \mathbf{I},$$

$$y_j \geq 0, \quad \text{integer}. \tag{2}$$

where

$\mathbf{J}$ = the set of indices associated with allowed shifts.

$\mathbf{I}$ = the set of demand periods.

y_j = the number of employees assigned to the jth shift.

c_j = the cost of an employee assigned to the jth shift.

$a_{ij} = \begin{cases} 1 & \text{if period } i \text{ is a work period in shift } j, \\ 0 & \text{otherwise,} \end{cases}$

r_i = the labor requirement in period i.

Note that formulation P1 appropriately models applications where the break is one or more planning (demand) periods in duration.

2.2. *Definitions*

The following terms are essential for model development:

(1) *Planning Period*—The smallest time interval for which forecasts of labor requirements have been established (typically 15 minutes to one hour). The use of the term "planning period" can be further clarified by noting that the operational day typically consists of a number of contiguous planning periods of equal duration.

(2) *Shift*—The detailed specification of the planning periods to be allocated to work and rest respectively.

(3) *Work Span*—The contiguous set of planning periods within which either work or rest must take place for a given shift.

(4) *Shift Length*—The total number of planning periods assigned for breaks as well as work for a given shift.

(5) *Shift Type*—The set of allowed shifts with the same work span.

(6) *Break Window*—The set of contiguous planning periods within which a break may begin for one or more work spans.

(7) *Extraordinary Overlap*—The existence of the following condition: The initial and final periods in the break window for any one shift type are earlier and later respectively, than the corresponding initial and final periods in the break window for at least one other shift type.

Note that a shift type is created by association of one or more break windows with a specific work span.

2.3. *Initial Assumptions*

We begin the development of the new model with the following initial assumptions:

(1) The organization operates less than 24 hours per day.

(2) All planning periods are of equal length.

(3) Every employee will receive exactly one break.

(4) The duration of breaks is identical for all shift types modeled.

(5) The duration of breaks is one or more planning periods.

(6) Each shift type is defined by the assignment of a single break window to an associated work span.

(7) The break window for each shift type consists of any selected nonzero subset of periods subject to the restriction that all periods contained within the break interval must be a subset of the work span associated with the shift type.

(8) Extraordinary overlap does not exist.

(9) No understaffing is allowed.

These assumptions do not present a severe limitation upon the more general application of an implicit modeling approach for a number of reasons. Numerous service organizations exist which do not operate on a continuous basis. Assumptions two and four have been incorporated in the vast majority of published applications and research in labor scheduling.

Assumption 8 must hold in order to guarantee that the resulting placement of breaks will be within the prescribed break window for all shifts. However, this assumption does not appear to present a serious issue for three reasons. First, we demonstrate in a later section of this paper that the implicit model appears to be capable of generating the desired optimal solutions even when a substantial degree of extraordinary overlap exists. Second, extraordinary overlap cannot exist unless a break window for one shift is at least two periods longer than the break window for a second shift. Finally, the existence of this latter condition does not necessarily result in extraordinary overlap.

As opposed to being restrictive, assumptions 5, 6, and 7 collectively allow for considerable flexibility in the duration and placement of breaks. Relaxations of assumptions 3 and 9 can be modeled with only minor modifications in the model to be presented in the next section.

2.4. *Implicit Formulation* (P2)

The implicit formulation (P2), is

$$\text{minimize} \quad \sum_{j \in \mathbf{T}} c_j x_j, \tag{3}$$

$$\text{s.t.} \quad \sum \mathbf{X_i} - \sum \mathbf{\Omega_i} \geq r_i \quad \text{for all} \quad i \in \mathbf{P}, \tag{4}$$

$$\sum \mathbf{F_k} - \sum \mathbf{G_k} \geq 0 \quad \text{for all but the last element} \quad k \in \mathbf{N}, \tag{5}$$

$$\sum \mathbf{R_k} - \sum \mathbf{S_k} \geq 0 \quad \text{for all but the first element} \quad k \in \mathbf{M}, \tag{6}$$

$$\sum \mathbf{X} - \sum \mathbf{B} = 0, \tag{7}$$

$$x_j \in \mathbf{X}, \qquad b_i \in \mathbf{B} \geq 0, \qquad \text{integer},$$

where

$\sum \mathbf{A}$ = the sum of all elements contained in any set $\mathbf{A}$,

$\mathbf{T}$ = the set of indices associated with allowed shift types,

c_j = the labor cost associated with one employee working the jth shift type,

x_j = the number of employees scheduled to work the jth shift type,

$\mathbf{X_i} = \{x_j | i \in \mathbf{W_j}\}$,

$\mathbf{W_j}$ = the work span associated with the jth shift type,

$\mathbf{X} = \{x_j | j \in \mathbf{T}\}$,

$\mathbf{\Omega_i}$ = the set of break variables for which period i is a break period,

r_i = labor requirement in period i,

$\mathbf{P}$ = the set of planning periods in the operating day,

$\mathbf{M}$ = the set of initial (earliest) periods (ascending order) in the break windows associated with all shift types,

$\mathbf{N}$ = the set of final (latest) periods (ascending order) in the break windows associated with all shift types,

p = the earliest period in the set of periods within which the break for any shift may begin,

q = the latest period in the set of periods within which the break for any shift may begin,

$\mathbf{U_j} = \{b_i | i \in \mathbf{V_j}\}$,

b_i = the total number of breaks initiated at the start of period i by the complete set of employees from all shift types,

$\mathbf{V}_j$ = the set of periods associated with the break window for the jth shift type,
$\mathbf{F}_k = \{b_i | k \in \mathbf{N} \text{ and } i = p, p+1 \cdots k\}$,
$\mathbf{G}_k = \{x_j | \mathbf{U}_j \subseteq \mathbf{F}_k\}$,
$\mathbf{R}_k = \{b_k | k \in \mathbf{M} \text{ and } i = k, k+1 \cdots q\}$,
$\mathbf{S}_k = \{x_j | \mathbf{U}_j \subseteq \mathbf{R}_k\}$,
$\mathbf{B} = \{b_i | i \in \mathbf{P}_b\}$,
$\mathbf{P}_b = \{p, p+1, p+2 \cdots q-1, q\}$.

Constraint set 4 ensures that demand requirements are met. Constraint sets 5–7 can be thought of as control constraints which are included to ensure that the shifts implicitly represented in formulation P2 are the explicitly represented shifts in formulation P1. Specifically, constraint set 5 is a "forward pass" set which ensures that sufficient quantities of breaks are appropriately available for allocation to subsets of shift types with break windows which are fully contained within the successively larger intervals $(p, p+1, \ldots, k)$ with $k \in \mathbf{N}$. The function of constraint set 6, the "backward pass" set, is similar to that of constraint set 5 except that the successively larger intervals are $(k, k+1, \cdots q)$ with $k \in \mathbf{M}$. Note that the intervals for constraint set 5 always include period p while the intervals associated with constraint set 6 always include period q. Taken together, constraint sets 5 and 6 ensure that each of the shifts represented in the optimal solution can be assigned a break which is within its prescribed break window. Constraint 7 ensures that exactly one break is available for each employee scheduled.

A proof of the equivalence of P1 and P2 is contained in Bechtold and Jacobs (1989) which was based upon the assumptions in §2.3. In the context of this paper, the term equivalence is used to indicate that the two models represent the same set of allowed shifts and that the optimal objective function values for the two models are equal.

Defining A as the number of elements in any set $\mathbf{A}$, the numbers of constraints (NC) and variables (NV) in P1 and P2 are

$$\text{NC(P1)} = P, \tag{8}$$

$$\text{NV(P1)} = \sum_{j \in \mathbf{T}} V_j, \tag{9}$$

$$\text{NC(P2)} = P + M + N - 1, \tag{10}$$

$$\text{NV(P2)} = T + B. \tag{11}$$

(10) provides further insight into the development of the control constraints. Specifically, a forward pass constraint is included for all periods in $\mathbf{N}$ except the latest period. A backward pass constraint is included for all periods in $\mathbf{M}$ except the earliest period. Thus, there are $N-1$ forward pass constraints, $M-1$ backward pass constraints, and 1 equality constraint for a total of $M+N-1$ control constraints.

In a general sense, the actual number of required control constraints is dependent upon the structure which results from the interaction effects of: (1) allowed shift lengths, (2) placement of shifts, (3) break window design, and (4) placement of break windows within shifts. However, if a shift is allowed to start in any period in which its length does not result in scheduled work beyond the latest period in the operational day and all break windows are contiguous, the total number of control constraints in P2 is

$$\text{NCC(P2)} = 2(P - D) + 1, \tag{12}$$

where D = the duration of the shortest allowed shift length. Thus, when equation (12) applies for a given operational day, the number of control constraints is determined solely by the duration of the shortest allowed shift length. This implies that the number of required control constraints is unaffected by the number of different shift lengths modeled.

If break flexibility is not being modeled, then NC(P2) > NC(P1) and NV(P2)

> NV(P1), and there is no general advantage in using P2. However, the relative advantage in using formulation P2 increases dramatically with increases in V_j, $j \in \mathbf{V_j}$ and increases in T. In many scheduling applications, T is a function of both P and the number of allowed shifts. Specifically, shifts are typically allowed to begin in all planning periods except for those which would result in scheduled work beyond the end of the operating day. If we assume that this latter condition holds, and define α = the number of different shift lengths allowed and l_k = the length of the kth shift, then $T = \sum_{k=1}^{\alpha} (P - l_k + 1)$.

The equality constraint in P2 may be exploited whenever the variable density of any constraint in set 5 or 6 is greater than half of the total variables ($T + B$). Such constraints in set 5 can be replaced with their corresponding complementary constraints

$$\sum \bar{\mathbf{F}}_k - \sum \bar{\mathbf{G}}_k \leq 0. \tag{13}$$

Similarly, high density constraints in set 6 may be replaced with their corresponding complementary constraints

$$\sum \bar{\mathbf{R}}_k - \sum \bar{\mathbf{S}}_k \leq 0. \tag{14}$$

2.5. *Break Allocation with* P2

The computational advantage inherent in P2 is conceptually achieved by requiring the model to produce less than the complete set of information required for the determination of actual work schedules and planned break assignments associated with the optimal work force. Specifically, the breaks determined by P2 (b_i^*, $i = p$ to q) must be assigned to the employees represented by shift type variables x_j^*, $j = 1$ to T. A simple and efficient procedure for accomplishing this allocation is:

(1) Set $b_i = b_i^*$ for $i = p$ to q, and $x_j = x_j^*$ for $j = 1$ to T.
(2) Set $i = p - 1$.
(3) Set $i = i + 1$.
(4) If $b_i = 0$ and $i < q$, go to 3. If $b_i = 0$ and $i = q$, stop. Otherwise, go to 5.
(5) Define $\mathbf{Z} = \{x_j | x_j > 0 \text{ and } i \in \mathbf{V_j}\}$ with elements x_j arranged in nondecreasing order according to the criterion of the latest period within $\mathbf{V_j}$. Ties are broken arbitrarily.
(6) Define x_j' = the lowest ranked element of $\mathbf{Z}$.
(7) Set a = min $\{b_i, x_j'\}$, $b_i = b_i - a$, and $x_j' = x_j' - a$.
(8) If $b_i = 0$, go to 3. If $x_j' = 0$, go to 6.

If P2 modeling assumptions are met, all such assignments are guaranteed to be within the domain associated with the shift decision variables represented in the corresponding formulation P1.

3. Experimental Analysis

This section provides a preliminary comparison of formulations P1 and P2 in a hypothetical work environment which is described in §3.1. In §3.2 we examine the CPU execution times of P1 and P2 which were associated with the solution of 40 test problems based upon all combinations of four labor requirements (demand) patterns and ten sets of allowed shift lengths. §3.3 provides a comparison of problem sizes and memory requirements for P1 and P2 across the same 40 test problems. §3.4 presents the results of a preliminary analysis of the equivalence of P2 schedule assignments to those allowed by P1 when assumption 8 was violated.

All computer analyses included in this section were executed on a CDC CYBER 850 computer. All formulations of P1 and P2 were executed using the GOMORY cutting plane option in MPOS (Cohen and Stein 1978). Complementary constraints, as defined by equations (13) and (14), were used in the execution of P2 whenever they resulted in

lower A-matrix density. The break allocation procedure described in §2.5 was programmed in FORTRAN 77 and executed for each of the 40 MPOS solutions obtained for formulation P2.

3.1. *Operational Environment*

For purposes of analysis, we consider the task of shift scheduling for a hypothetical organization which has previously determined the labor requirements (demand) for an operational day composed of 24 consecutive planning periods. The actual length of a planning period may be assumed to be any convenient duration which is less than one hour.

We consider four different labor requirements patterns in the subsequent analyses. All demand patterns were set with a mean demand of 25 persons/period with a total demand of 600 across the 24 planning periods. Minimum and maximum demands across the four patterns were 5 and 45 respectively. The first pattern (Demand 1) was unimodal across the work day with lowest labor requirements in the initial and final periods. The remaining patterns (Demand 2 through Demand 4) were all sinusoidal with one, two, and three planning period(s) with peak requirements respectively.

The selection of shift-length combinations to generate relative performance characteristics for the two modeling approaches was based upon the use of reasonable minimum and maximum shift lengths while simultaneously varying the size of the test problems. The measure of problem size used in this selection process was the required number of set-covering variables. The ten specific shift-length combinations used are presented in Table 1 in ascending order with respect to the required number of set-covering variables. The total number of different shifts represented by each of the ten shift-length combinations was based upon the assumption that a shift of any allowed length could begin in any period which did not result in scheduled work beyond the end of the operational day.

All shifts included a one-period break. As in Byrne and Potts (1973) the break window lengths were selected such that no break would occur in the first or last period of the work spans used. This resulted in respective meal break windows consisting of 7 and 8 periods for the 9 and 10 period work spans. In order to obtain increases in problem size with increasing numbers of allowed shift lengths, break placement for all shifts was prescribed by: (1) centering a seven-period break window on work spans with odd durations, and (2) centering an eight-period break window on work spans with even durations. Although these break windows allow for a relatively high degree of break-placement flexibility, the overall scheduling flexibility represented is relatively low with shift-length combination 1 and relatively high with shift-length combination 10.

3.2. *Comparison of CPU Times*

Matrix generation, input, and output (M/I/O) times for P1 and P2 on the ten shift-length combinations are displayed in Table 1. For a given model, these times were found to be identical (within hundredths of a CPU second) across the four labor requirements patterns. Model P1 exceeded the memory capacity of the computer installation for the last six shift-length combinations. Therefore, the associated output times for P1 are not displayed in Table 1.

An examination of Table 1 indicates that P1 required more matrix generation, input, and output time than P2 for every test problem for which comparative data was obtained. Based upon the raw data, the minimum and maximum P1/P2 ratios across all comparable pairs of matrix generation, input, and output times were 2.20 and 4.63 respectively. In general, the times required by both P1 and P2 were an increasing function of the required

TABLE 1

Matrix Generation and Input/Output Times for P1 *and* P2[1]

Shift-Length Combination	Available Shift Lengths	Number of Variables		Matrix Generation Time (M)		Input Time (I)		Output Time (O)		Total Time (M/I/O)	
		P1	*P2*	*P1*	*P2*	*P1*	*P2*	*P1*	*P2*	*P1*	*P2*
1	15, 17	126	34	1.32	.54	.84	.27	.11	.05	2.27	.86
2	13, 15	154	40	1.41	.61	.91	.32	.13	.05	2.45	.98
3	15, 17, 19	168	40	1.79	.63	1.16	.33	.15	.06	3.10	1.02
4	15, 17, 19, 21	196	44	2.15	.68	1.40	.37	.19	.06	3.74	1.11
5	13, 15, 17	210	48	1.98	.72	1.30	.39	—	.07	—	1.18
6	11, 13, 15	252	56	2.07	.81	1.38	.45	—	.07	—	1.33
7	11, 13, 15, 17, 19	350	70	3.09	1.02	2.12	.58	—	.07	—	1.67
8	9, 11, 13, 15	364	74	2.66	1.03	1.83	.59	—	.09	—	1.71
9	9, 11, 13, 15, 17, 19, 21	490	92	4.03	1.31	2.82	.76	—	.10	—	2.17
10	9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21	970	152	7.93	2.09	5.93	1.28	—	.18	—	3.55

[1] All times reported are CPU seconds on a Cyber 850 computer.

number of set-covering variables with the time required by P1 growing at a faster rate than for P2.

Table 2 displays the mean, minimum, and maximum integer programming solution (S) times for P1 and P2 across the four labor requirements patterns. The solution time measures for P2 included the CPU time required for break allocation. The break allocation

TABLE 2

Integer Programming Solution Time and Total CPU Time for P1 *and* P2[1]

Shift-Length Combination	Available Shift Lengths	Mean Solution Time (S)		Minimum Solution Time		Maximum Solution Time		Mean Total Time (M/I/S/O)	
		P1	P2	P1	P2	P1	P2	P1	P2
1	15, 17	.86	.17	.39	.11	1.71	.30	3.13	1.03
2	13, 15	2.20	.74	.81	.17	5.04	2.25	4.65	1.72
3	15, 17, 19	1.23	.22	.42	.11	2.46	.35	4.33	1.24
4	15, 17, 19, 21	1.59	.26	.53	.14	3.07	.39	5.33	1.37
5	13, 15, 17	—[2]	.84	—	.19	—	2.60	—	2.02
6	11, 13, 15	—	.81	—	.27	—	2.22	—	2.14
7	11, 13, 15, 17, 19	—	1.22	—	.34	—	3.72	—	2.89
8	9, 11, 13, 15	—	1.58	—	.59	—	3.94	—	3.29
9	9, 11, 13, 15, 17, 19, 21	—	2.33	—	.81	—	5.97	—	4.50
10	9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21	—	3.54	—	1.56	—	7.63	—	7.09

[1] All times reported are CPU seconds on a Cyber 850 computer.

[2] Problem size for P1 for the last six shift-length combinations exceeded memory limits, and no solution times were available.

times ranged from a minimum of .003 seconds for shift-length combination 1 to a maximum of .141 seconds for shift-length combination 10. Based upon the raw data, the mean of the ratio of break allocation time to total P2 solution time was approximately 1.4 percent.

Again, in every instance where comparative data was available, the solution times for P1 were always greater than the corresponding times for P2 across the three time measures displayed in Table 2. The P1/P2 solution time ratios across the four labor requirements patterns for the four comparable shift-length combinations ranged from a minimum of 2.24 to a maximum of 7.26. The mean solution times appeared to be increasing more rapidly for P1 than for P2 with increases in problem size. Given the lack of solution time data for P1 on the last six test problems, the support for this observation was not as strong as it was for the corresponding observation with respect to Table 1.

Table 3 displays a summary of P1/P2 time ratios for: (1) total M/I/O time, (2) mean solution times, and (3) the sum of M/I/O and mean solutions times. These ratios provide further support for the increasing advantage of using P2 with larger problems. Table 3 also indicates that the mean solution time ratios were always larger than the ratios for total M/I/O time.

3.3. *Comparison of Problem Size Characteristics*

Table 4 displays the number of constraints, the number of nonzero elements in the A-matrix, and the corresponding percentages of nonzero elements for both models across the ten shift-length combinations. For convenience, the total number of variables is also reproduced in Table 4.

Table 4 indicates that P1 always required more variables than P2. Moreover, a sequential examination of the results for shift-length combination 1 through 10 reveals that the number of required variables increased more rapidly for P1 than for P2.

The assumptions made in the development of allowed shifts in this experimental analysis match those which apply to equation (12). Therefore, the implicit model required a total of $3P - 2D + 1$ constraints. The fact that the number of constraints in the implicit model is unaffected by the number of shift lengths modeled is evident for all shift-length combinations in Table 4. For example, the minimum shift length in combination 4 is 15 periods and the total number of constraints is 43 ($3*24 - 2*15 + 1$).

Due to the common minimum shift length of 9 periods, the total number of constraints in the implicit model is the same for each of the last three shift-length combinations. Thus, while 31 control constraints are required to implicitly model a 9-period shift length, the 12 additional shift lengths in combination 10 can be modeled without a further increase in the number of constraints.

The number of nonzero elements associated with P2 was always substantially less than the corresponding number associated with P1. While the number of nonzero elements

TABLE 3

Computational Time Ratios for P1 *and* P2

Shift-Length Combination	Available Shift Lengths	Ratio P1/P2 (M/I/O Time)	Ratio P1/P2 (S Time)	Ratio P1/P2 (M/I/O/S Time)
1	15, 17	2.64	5.06	3.04
2	13, 15	2.50	3.00	2.70
3	15, 17, 19	3.04	5.60	3.49
4	15, 17, 19, 21	3.37	6.12	3.89

TABLE 4

Problem Size Characteristics for P1 and P2

Shift-Length Combination	Available Shift Lengths	Total Variables		Number of Constraints		Number of Nonzeroes in A-Matrix		Percent Nonzeroes in A-Matrix	
		P1	P2	P1	P2	P1	P2	P1	P2
1	15, 17	126	34	24	43	1876	516	62.0	35.3
2	13, 15	154	40	24	47	1988	618	53.8	32.9
3	15, 17, 19	168	40	24	43	2632	654	65.3	38.0
4	15, 17, 19, 21	196	44	24	43	3192	750	67.9	39.6
5	13, 15, 17	210	48	24	47	2884	794	57.2	35.2
6	11, 13, 15	252	56	24	51	2968	926	49.1	32.4
7	11, 13, 15, 17, 19	350	70	24	51	4620	1240	55.0	34.7
8	9, 11, 13, 15	364	74	24	55	3864	1260	44.2	31.0
9	9, 11, 13, 15, 17, 19, 21	490	92	24	55	6076	1670	51.7	33.0
10	9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21	970	152	24	55	12236	2004	52.6	34.7

increased for both P1 and P2 from the first to last shift-length combination, the rate of growth in the number of nonzero elements for P1 was approximately twice that of P2.

Table 4 also reveals that even though the size of the A-matrix for P2 was always smaller than the corresponding matrix for P1, the percentage of nonzero elements for P2 was always less than the corresponding percentage for P1. These percentages did not appear to be a function of differences in the number of set-covering variables. Overall, the nonzero density for P1 averaged approximately 25 percentage points above that of P2.

3.4. *Experimental Analysis of the Impact of Extraordinary Break-Window Overlap*

This section summarizes the results of a preliminary investigation of the robustness of formulation P2 when assumption eight was relaxed. The two break-window structures

TABLE 5

Two Break-Window Structures Used to Investigate the Impact of Extraordinary Break-Window Overlap

Shift Lengths	Break-Window Structure A			Break-Window Structure B		
	From	To	Number of Periods in Break Window	From	To	Number of Periods in Break Window
9	4	6	3	4	6	3
10	4	6	3	4	6	3
11	4	8	5	4	8	5
12	4	8	5	4	8	5
13	5	9	5	4	10	7
14	5	9	5	4	10	7
15	5	11	7	4	12	9
16	5	11	7	4	12	9
17	5	13	9	4	14	11
18	5	13	9	4	14	11
19	6	14	9	4	16	13
20	6	14	9	4	16	13
21	7	15	9	4	18	15

which were utilized are displayed in Table 5. The break windows in both structures were designed such that it was possible for extraordinary overlap to exist between any two shifts with lengths which differed by at least two periods. While both structures permit a high degree of extraordinary overlap, structure B allows a higher degree than structure A.

Structures A and B were executed with MPOS for all four demand patterns with all ten shift-length combinations in Table 1. This resulted in 80 runs made to examine the effects of extraordinary overlap. The application of the break allocation program to the 80 MPOS solutions resulted in the placement of all breaks within the limits set by the prescribed windows.

4. Summary and Conclusions

A new implicit integer programming formulation was developed in this paper which allows scheduling flexibility to be parsimoniously modeled in a shift scheduling environment. In order to compare the size and performance of the two models, a preliminary experimental analysis was conducted using 40 test problems. Differences in scheduling flexibility across the test problems were generated by increasing the number of allowed shifts while holding the degree of break-placement flexibility constant. The test problems were based upon all combinations of four different labor requirements patterns with ten different sets of allowed shift lengths. Integer optimal solutions were obtained for all 40 test problems using the implicit formulation with commercially available software. The set-covering model would not execute on any problem which required over 200 variables, and thus, integer optimal solutions were obtained for 16 test problems.

Across all problems which resulted in comparable data, the implicit formulation required less CPU time in each of four categories: (1) matrix generation (M), (2) input to integer programming software (I), (3) integer programming solution (S), and (4) output (O). The P1/P2 time ratios for M/I/O ranged from 2.64 to 3.37 across the 16 problems for which both models executed. The mean P1/P2 ratios for integer programming solution time across the same 16 problems ranged from 3.00 to 6.12. In general, total solution time for either model was an increasing function of the number of shift lengths modeled; the rate of increase was higher for the set-covering model than for the implicit model.

The size of the A-matrix required for the implicit model was substantially smaller than the corresponding size for the set-covering model for all ten shift-length combinations. The nonzero A-matrix density for the set-covering model was always higher than that for the implicit model with an average difference of about 25 percentage points.

In order to control the distribution of breaks, the implicit model requires more constraints than the set-covering model. However, the number of control constraints in a typical application for a given operational day is determined solely by the duration of the shortest shift to be modeled. Any number of longer shift lengths which are compatible with the duration of the operational day may be modeled without an additional increase in the number of control constraints. Alternatively, when break-placement flexibility is modeled, the number of variables in the set-covering model tend to increase at a much faster rate for the set-covering model than for the implicit model. Thus, for one of the largest problems in the experimental analysis, the implicit model required 818 fewer variables at the expense of 31 additional constraints.

A secondary experimental analysis of the impact of extraordinary break-window overlap across shifts was also conducted. The results revealed that the actual placement of breaks was always within the prescribed break window for all shifts for all 80 test problems. Thus, the implicit modeling approach appeared to be robust with respect to its capability to produce the desired optimal work schedules.

These latter results may be partially explained by the fact that the break window for one shift must be at least two periods longer than that for an alternative shift for extraordinary overlap to exist. While increasing the size of a break window may create extraordinary overlap, it also increases the number of break assignment alternatives across the overlapped shifts. Moreover, if extraordinary overlap does exist between two shifts, the break assignment procedure developed in this paper will assign a break to the shift with the shorter break window before doing so for the alternative shift.

Numerous research opportunities exist for the extension of the implicit modeling approach for use in other operating environments. For example, shift scheduling for telephone operators is typically done across 24 hour planning horizons. Moreover, shift scheduling environments exist in practice for which the organization desires to include optimal placement of rest breaks as well as meal breaks in shift assignments (Henderson and Berry 1976).

The extension of implicit modeling to tour scheduling environments would appear to be very worthwhile due to the fact that tour scheduling problems are generally quite large relative to shift scheduling problems. This is due, in part, to the fact that daily shift assignments are integral to the task of solving the tour problem which frequently has a planning horizon of a week or longer. Given the potential CPU time savings which could be expected along with reduced A-matrix size, it may be possible to obtain optimal solutions to large tour scheduling problems using an implicit modeling approach.

All computational results reported in this paper were based upon the use of the all-integer cutting plane procedure available in MPOS. Further research using mixed-integer cutting plane and/or branch and bound algorithms should be conducted. The experimental analysis conducted here was limited in part by space considerations, and it may be worthwhile to investigate the performance characteristics of the two models across a broader range of break-placement and shift-length flexibility.

Finally, the availability of an implicit modeling approach should encourage more research into the potential benefits of scheduling flexibility. To date, only two studies have been published in this regard (Bailey and Field 1985; Showalter and Mabert 1988). Due to problem size, the results of these two studies were based upon heuristic solution procedures.[1]

[1] We wish to express our appreciation to the editors and the anonymous referees for their suggestions which resulted in a substantial improvement in the exposition of this paper.

References

BAILEY, J. AND J. FIELD, "Personnel Scheduling With Flexshift Models," *J. Operations Management*, 5, 3 (1985), 327–338.

BECHTOLD, S. E. AND L. W. JACOBS, "The Equivalence of Set-Covering and Implicit Optimal Integer Programming Formulations for Mathematical Labor Scheduling Problems," Working Paper, Florida State University, 1989.

BYRNE, J. L. AND R. B. POTTS, "Scheduling of Toll Collectors," *Transportation Sci.*, 3 (1973), 224–245.

COHEN, C. AND J. STEIN, *Multi-Purpose Optimization System User's Guide*, version 4, Vogelback Computing Center, Northwestern University, Evanston; (1978).

DANTZIG, G., "A Comment on Edie's Traffic Delay at Toll Booths," *Oper. Res.*, 2, 3 (1954), 339–341.

GABALLA, A. AND W. PEARCE, "Telephone Sales Manpower Planning at Qantas," *Interfaces*, 9, 3 (1979), 1–9.

GLOVER, F., C. MCMILLAN AND R. GLOVER, "A Heuristic Programming Approach to the Employee Scheduling Problem and Some Thoughts on 'Managerial Robots,'" *J. Operations Management*, 4 (1984), 113–128.

——— AND ———, "The General Employee Scheduling Problem: An Integration of MS and AI," *Computers and Oper. Res.*, 13, 5 (1986), 563–573.

HENDERSON, W. B. AND W. L. BERRY, "Heuristic Methods for Telephone Operator Shift Scheduling: An Experimental Analysis," *Management Sci.*, 22, 12 (1976), 1372–1380.

HOLLORAN, T. J. AND J. E. BYRN, "United Airlines Station Manpower Planning System," *Interfaces*, 16, 1 (1986), 39–50.

KEITH, E. G., "Operator Scheduling," *AIIE Trans.*, (March 1979), 37–41.

MOONDRA, S. L., "An L.P. Model for Work Force Scheduling for Banks," *J. Bank Res.*, (Winter 1976), 299–301.

SEGAL, M., "The Operator-Scheduling Problem: A Network Flow Approach," *Oper. Res.*, 22 (1974), 808–823.

SHOWALTER, M. J. AND V. A. MABERT, "An Evaluation of A Full-/Part-time Tour Scheduling Methodology," *Internat. J. Operations and Production Management*, 8, 7 (1988), 54–71.

TAYLOR, P. E. AND S. J. HUXLEY, "A Break From Tradition for the San Francisco Police: Patrol Officer Scheduling Using an Optimization-Based Decision Support System," *Interfaces*, 19, 1 (1989), 4–24.

THOMPSON, G. M., "A Comparison of Techniques for Scheduling Non-Homogenous Employees in a Service Environment Subject to Non-Cyclical Demand," Unpublished Ph.D. Dissertation, Florida State University, 1988.

Comput. & Indus. Engng Vol. 9, No. 4, pp. 395–404, 1985
Printed in the U.S.A.

0360-8352/85 $3.00 + .00

INTEGRATED DAYS OFF AND SHIFT PERSONNEL SCHEDULING

JAMES BAILEY
Arizona State University, Tempe, AZ 85287, U.S.A.

(*Received for publication* 12 *March* 1985)

Abstract—The two levels of personnel scheduling, that is, determination of the days an employee should work and determination of the time an employee should start each workday, are integrated. The integration allows constraints involving hourly fluctuations in demand and fixed work force size to interact. The objective is to minimize the cost of premium pay plus customer inconvenience due to understaffing. An optimal solution is attained via linear programming. In a realistic service system environment, the model outperformed an integrating heuristic by 13.56%.

INTRODUCTION

In service organizations, output cannot be inventoried. The demand for service requires the effort of an employee promptly upon the customer's arrival. As a result, a significant problem in organizations such as telephone exchanges, restaurants, retail stores, hospitals, police, and fire departments is one of scheduling personnel to meet fluctuating customer demands.

The personnel scheduling problem has traditionally been solved at two separate but related levels. The "shift-scheduling" problem is to identify the number of 8-hour shifts needed to satisfy fluctuating hourly demand over the course of the day. The "days-off" problem is to determine the number of people needed to satisfy the daily demands while guaranteeing adequate days off. Clearly these problems are related. The solution to the "shift-scheduling" problem is the input to the "days-off" problem. In the reverse direction, staff size and overtime constraints in the "days-off" problem limit the availability of people in the "shift-scheduling" problem. One attempt at combining these two levels of personnel scheduling is referred to as the "tour" problem where five 8-hour shifts with identical start times are scheduled into 168 hrs of fluctuating demand. It is the intent of this paper to present a more robust solution to the integrated personnel scheduling problem. The solution is formed as a decomposable linear program with unique exploitable properties.

LITERATURE SURVEY OF PERSONNEL SCHEDULING

Models of the "shift-scheduling" problem have appeared in a number of papers. Baker[1] presents a detailed review of many of these techniques. Luce[2, 3] developed a heuristic to select shifts, one at a time, from a "working subset" of acceptable shift patterns. The selection criterion was based on the proportion of unsatisfied demand met by each candidate pattern. Henderson and Berry[4] presented three heuristics for identifying good "working subsets" from all possible shift patterns. With these acceptable shifts, they presented heuristics based on L.P. and a neighborhood search technique for scheduling shifts. Segal[5] describes a network flow model for optimal shift selection. Keith[6] presented an L.P. formulation which minimized the cost of understaffing and overstaffing. Baker[1] also suggests L.P. formulations that allowed limited deviation from the demand function in both the understaffing and overstaffing directions. Koelling[7] presented a goal programming model to include secondary objectives such as overstaffing and understaffing. Field[8] presented an L.P. model that considered flexible shifts of differing duration.

The solution to the "days-off" problem has also received considerable attention. A simple heuristic to select work day patterns with two consecutive days off was offered

by Tibrewala *et al.*[9]. Monroe[10] presented a heuristic which did not require consecutive days off. Rothstein[11] presented an L.P. formulation which maximized the number of shifts with two consecutive days off but accepted shifts with two nonconsecutive days off. This solution was constrained to exactly satisfy demand on each day.

McGinnis *et al.*[12] presented heuristics to solve the "tour" problem. In his one-phase heuristic, the intent was to schedule five 8-hour shifts with identical start times based on a simple variation of the Luce model mentioned above. These results are presented in greater detail because they are used later for comparison purposes. Work patterns were scheduled, one at a time, by calculating a priority index for each feasible work pattern and choosing the pattern with the maximum index. This was continued until the total work force was scheduled. The index for a feasible schedule was given by:

$$P_{ij} = \begin{cases} (1 - S_{ij}/D_{ij})^2 & \text{if } D_{ij} \geq S_{ij} \\ -(1 - S_{ij}/D_{ij})^2 & \text{if } D_{ij} < S_{ij} \\ 0 & \text{if } D_{ij} = 0 \end{cases}$$

$$P_k = \sum_{T_k} P_{ij},$$

where

P_k = Priority index for pattern k summed over all periods T_k in which pattern k is working,
P_{ij} = Priority index of hour i on day j,
S_{ij} = Supply of people already scheduled to work in hour i of day j, and
D_{ij} = Demand for people to work in hour i of day j,

INTEGRATED MODEL FORMULATION

The "days-off" and "shift-scheduling" problems need to be integrated because their constraints interact. To understand this integration consider first the "shift-scheduling" problem. Utilizing Dantzig's[13] set covering formulation, Baker[14] proposed the following simple model of this problem

$$\text{Min } Z_i = \sum_{k=1}^{K_i} C_{ik}x_{ik}, \tag{1}$$

$$\text{S. T.} \quad \sum_{k=1}^{K_i} A_{lk}x_{ik} \geq D_{il}; \quad \text{for } l = 1, \ldots L_i, \tag{2}$$

$$x_{ik} \geq 0 \text{ and integer}, \tag{3}$$

where i = Monday, Tuesday, . . . , Sunday, where:

x_{ik} = number of people scheduled to work shift pattern k on day i.
C_{ik} = labor cost of pattern k on day i,
A_{lk} = an $(L_i \times K_i)$ matrix where $a_{lk} = 1$ if pattern k is working during period l and $a_{lk} = 0$ otherwise,
D_{il} = demand for people during period l of day i,
K_i = number of feasible patterns on day i, and
L_i = number of periods to be scheduled on day i.

The above model is constrained to meet or exceed the demand during each period. It could also have been constrained to a known work force w_i by introducing slack and surplus variables into each period and adding a new constraint that requires the number of shifts to equal the available people. The slack variable should also be included in

the objective function to account for customer inconvenience cost when understaffed. Note that if there is no difference in labor cost between shift patterns, labor cost is constant and $C_{ik}w_i$ can be removed from the objective function. This leads to the following model:

$$\text{Min } Z_i = \sum_{l=1}^{L_i} \alpha_{il} U_{il}, \tag{4}$$

$$\text{S. T. } \sum_{k=1}^{K_i} A_{lk}x_{ik} - O_{il} + U_{il} = D_{il}; \text{ for } l = 1, \ldots L_i, \tag{5}$$

$$\sum_{k=1}^{K_i} x_{ik} = w_i, \tag{6}$$

$$x_{ik} \geq 0 \text{ and integer}, \tag{7}$$

where:

O_{il} = overstaffing surplus during period l of day i,
U_{il} = understaffing slack,
α_{il} = customer inconvenience cost per worker due to understaffing, and
w_i = available supply of workers on day i.

Consider now a similar formulation of the days-off problem. In this case, a feasible work pattern consists of certain days-on and days-off. The cost vector reflects the labor costs for each feasible pattern. The demand vector is the total requirement for people during each day. The model is therefore:

$$\text{Min } Z = \sum_{j=1}^{J} C_j X_j, \tag{8}$$

$$\text{S.T. } \sum_{j=1}^{J} A_{ij}X_j \geq D_i; \text{ for } i = 1, \ldots 7, \tag{9}$$

$$X_j \geq 0 \text{ and integer}, \tag{10}$$

where:

X_j = the number of people working days-off pattern j,
C_j = cost of one person on pattern j,
A_{ij} = a $(7 \times J)$ matrix where $a_{ij} = 1$ if pattern j calls for working on day i and $a_{ij} = 0$ otherwise,
D_i = demand for workers on day i, and
J = number of feasible days-off work patterns.

In a limited staff size version of this formulation, the total size of the work force causes another constraint. The sum of the work patterns must in this case equal the size of the work force. The formulation therefore becomes:

$$\text{Min } Z = \sum_{j=1}^{J} C_j X_j + \sum_{i=1}^{7} \Gamma_i U_i, \tag{11}$$

$$\text{S.T. } \sum_{j=1}^{J} A_{ij}X_j - O_i + U_i = D_i, \tag{12}$$

$$\sum_{j=1}^{J} X_j = W_{\text{total}}, \tag{13}$$

$$X_j \geq 0 \text{ and integer}, \tag{14}$$

where:

W_{total} = total size of the work force,
Γ_i = customer inconvenience cost of being understaffed by one person on day i,
U_i = understaffing during day i, and
O_i = overstaffing during day i.

Once again, if C_j is constant for all X_j the first portion of the objective function can be dropped. We do not do so here because premiums paid for overtime, Saturdays and Sundays suggest a nonconstant value for C_j.

The "shift-scheduling" formulation, (eqns (4–7)) and "days-off" formulation (eqns (8–10)) are integrated in three ways. First the objective functions are combined by adding seven replications of eqn (4) (one for each day) to eqn (8). Second, eqns (6) and (9) are combined by noting that the available supply of workers w_i is equal to the demand for workers D_i. The third requirement is to add a new constraint limiting the total work force to W_{total} people. The integrated formulation thereby becomes:

$$\text{Min } Z = \sum_{j=1}^{J} C_j X_j + \sum_{i=1}^{7} \sum_{l=1}^{L_i} \alpha_{il} U_{il}, \tag{15}$$

$$\text{S.T.} \quad \sum_{j=1}^{J} A_{ij} X_j - \sum_{k=1}^{K_i} x_{ik} = 0; \qquad \text{for } i = 1, \ldots 7, \tag{16}$$

$$\sum_{k=1}^{K_i} A_{lk} x_{ik} - O_{ik} + U_{il} = D_{il} \qquad \text{for } l = 1, \ldots L_i \text{ and } i = 1, \ldots 7, \tag{17}$$

$$\sum_{j=1}^{J} X_j = W_{\text{total}}, \tag{18}$$

$$x_{ik}, X_j \geq 0 \text{ and integer}, \tag{19}$$

where:

W_{total} = total size of the work force.

Note that the objective of the integrated problem is to minimize the weekly labor cost of W_{total} people plus the sum of the hourly customer inconvenience cost due to understaffing U_{il}. Constraint eqn (16) requires the number of shifts X_j supplied by the days off problem be equal to the number of shifts x_{ik} used in each of the shift scheduling problems. Constraint eqn (17) requires that the personnel scheduled during each period of each day is equal to the demand when overstaffing and understaffing are accounted for. Constraint eqn (18) limits the shifts to the total available work force. Thus this formulation attempts to schedule a fixed work force to meet hourly demand over seven days. The formulation allows for five, six and seven day work weeks with consecutive or nonconsecutive days off. The cost of alternative work patterns is traded off against the organizational cost incurred due to customer inconvenience when too few people are scheduled. The X_j portion of the solution vector indicates the days-on/days-off patterns that the workers can select from. The x_{ik} portion of that vector indicates the required shift patterns that are required for each day. The feasible set of shift patterns and the number of periods can easily be varied for each day. Using additional constraints, specific patterns can be guaranteed or limited to reflect personnel preferences. The shadow price on eqn (18) indicates the change in overall cost resulting from an increase or decrease in staff size. The model appears to be quite flexible.

To further understand the formulation, the A matrix is represented in Fig. 1. The

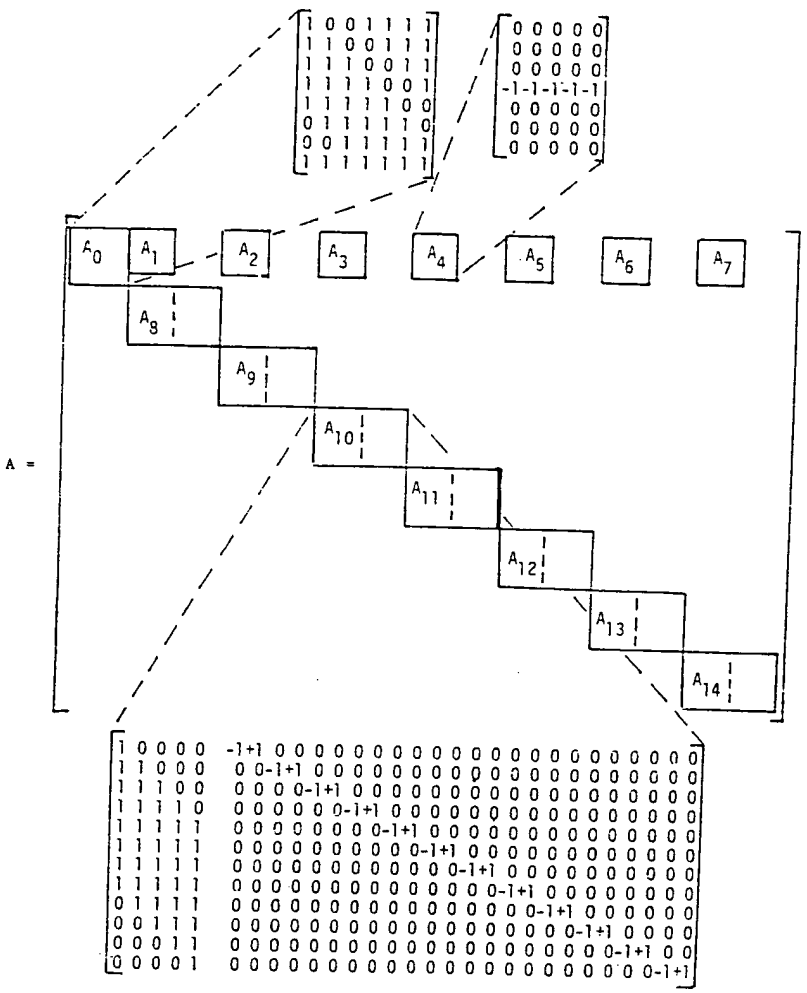


Fig. 1. Submatrices of the A matrix for the integrated model.

matrix is sparse and its nonzero portions can be divided into 15 submatrices. The $\mathbf{A}_0$ matrix in Fig. 1 represents the X_j portion of eqns (16) and (18). The M columns of $\mathbf{A}_0$ reflect the M feasible days-off pattern. Figure 1, for example, identifies seven patterns with two consecutive days off. The top seven rows in $\mathbf{A}_0$ represent the $\sum A_{ij}X_j$ portion of eqn (16) and the bottom row represents the $\sum X_j$ in eqn (18). The $\mathbf{A}_1$ through $\mathbf{A}_7$ submatrices represent the $\sum x_{ik}$ portions of eqn (16). These are essentially null matrices with a single row of negative ones. The $\mathbf{A}_8$ through $\mathbf{A}_{14}$ submatrices represent eqn (17). The L_i rows represent the hourly periods of the day. The left hand K_i columns represent all feasible shifts. For example, the five columns in the figure reflect five 8-hour shifts in a 12-hour work day. The right hand $2 \times L_i$ columns represent the overstaffing and understaffing portions of eqn (17). Thus the A matrix has $M + 7 \times (K_i + 2L_i)$ columns and $8 + 7 \times L_i$ rows. For Fig. 1, this represents a (92 × 210) matrix which can be handled even by microcomputer based linear programming software.

EXTENSION TO A 24-HOUR DAY

Figure 1 demonstrates the formulation for a 12-hour work day. The model can be adapted to work days of various lengths. However, when a 24-hour workday is considered, a complication arises. A worker who is scheduled to work on two consecutive days could be forced to work back-to-back shifts and not be given time for sufficient mental and physical rest. To guarantee a rest period, the model can be extended to three overlapping 12-hour periods covering a 24-hour day. These periods are shown in Fig. 2.

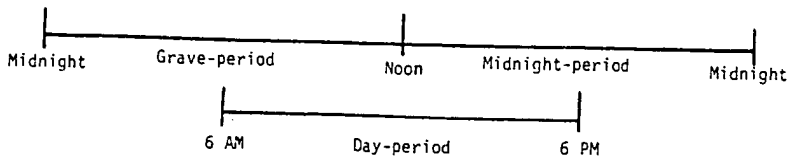


Fig. 2. Traditional shift periods for flexishift models.

By limiting the feasible work patterns in the A matrix to start and finish in the same 12-hour period and assigning each worker to the same period throughout the week, a minimum of 12 hrs between work periods is guaranteed. The higher (i.e. premium) cost on work patterns in the grave-shift and midnight-shift periods, causes the solution to select day-shift patterns whenever possible.

INTEGER SOLUTIONS

The formulation given in eqns (15–19) calls for integer solutions. The use of linear programming software would be much faster but may yield fractional results. This concern has been discussed by several authors[1, 8 and 15]. Bartholdi *et al.*[15] described a simple procedure that yields optimal integer results provided each row in the A matrix is a single string of consecutive ones within consecutive zeros. This holds for the "shift-scheduling" and "days-off" subproblems but not the integrated formulation. A simple expedient of rounding fractional results up is, however, available for the integrated problem. Experience in testing the integrated formulation using integer demand vectors and linear programming software always resulted in integer solutions.

SHIFT ASSIGNMENT HEURISTIC

The solution given by the L.P. formulation does not assign start times to the required days in each shift pattern. Therefore a shift assignment heuristic is needed. In order to reduce the occurrences of shift patterns with maximum difference in start times, the following heuristic is offered.

Step 1. From the L.P. result create a shift requirements table as shown in Figs. 3(a) and 3(b).

Step 2. Select the smallest shift requirement with a start time nearest the middle of the possible start times. Break ties arbitrarily. In Fig. 3(a) there are 38 shifts which must start at 8 am on Saturday.

Step 3. Identify the days-off patterns which require the shift selected in step two. In Fig. 3(b) there are shift patterns B, C, D and F.

Step 4. If more than one pattern is available select patterns with more days first. Break ties arbitrarily. In Fig. 3(b), this is pattern F which contains 6 days.

Step 5. Assign shifts to the selected pattern using the following criteria:
   a. assign shifts selected in step 2
   b. assign all other shifts from either the earliest or the latest available start times. In Fig. 3(a), these are 6 am and 10 am shifts.
   Reduce the shift requirements table to reflect the assignment.

Step 6. If all shifts are assigned, stop, Otherwise return to step 2.

COMPARISON OF MODELS

To study the efficacy of the integrated L.P. model given by eqns (15–19), comparisons were made to two alternative formulations. Those forms were the one phase "tour" heuristic suggested by McGinnis *et al.*[12] and a modification of the two phase optimization suggested by Baker[1] and Field[8]. The test data were six weeks worth of 12 hr days as given by McGinnis and based on operator requirements for a telephone company. Week 1 of the data represented actual demand while subsequent weeks represented successively smoothed versions of the actual data.

All formulations allowed for operators to work any five consecutive or six days in the week. All shifts were eight hrs long allowing for four feasible start times. By summing the demand for each week and dividing by 40 hrs/week, a target number of employees required for each week was estimated. This target number was assumed to represent the size of the work force. Labor costs were set at one dollar/hr. The cost of Saturdays, Sundays and a sixth overtime day were included using standard time and a half and double time considerations. The cost of customer inconvenience was somewhat arbitrarily set at 2.5 times the standard hourly wage so as to make understaffing more costly than even the double time wage for Sundays. The measure of performance was the sum of the labor cost and the customer inconvenience cost as given by eqn 11.

Three scheduling heuristics were tested. The integrated model given by eqns (15–19) was the first method. A second heuristic was the one-phase method offered by McGinnis and presented earlier in this paper. Finally a two step L.P. heuristic was tested. All tests utilized the same demand data and feasible shift patterns.

The two phase L.P. formulation was taken from models represented by eqns (4–7) and eqns (11–14). First the days-off problem was solved to determine the labor available for each day. The cost of understaffing required by eqn (11) was set at eight times the hourly understaffing cost since each person worked eight hrs/day. Once the daily staff sizes were determined, seven shift-scheduling problems were solved. The total cost of the solution was the sum of the labor cost from the days-off solution plus the understaffing cost from the seven shift-scheduling solutions.

The results of these tests are presented in Table 1. Comparisons are presented as total labor plus understaffing costs and as a percent savings of the integrated model over the two alternative solutions. On the average, the integrated model resulted in a cost savings of 13.56% of the solution given by the McGinnis one-phased method and 3.76% of the solution given by the two-step L.P. model. One could attribute this slight improvement over the two step L.P. method to the integration of eight optimization problems into one. The more significant improvement over the McGinnis method came, in part, from increased flexibility in shift selection.

The McGinnis one phase heuristic was limited to work patterns that started the same time during each day. This resulted in 56 feasible shifts. When multiple starts have been allowed, as in the integrated L.P. model, there are 35,840 feasible shifts which would be very costly to search each time a new shift was to be scheduled. The optimization nature of the integrated L.P. method must also account for some of the gain.

Table 2 contains additional comparison statistics for week 1 data using the three approaches. Several observations are possible. First, the two L.P. methods did a better

Table 1. Comparison of three formulations in terms of total cost

		Cost			Percent Savings to	
Week	Target Work Force	(A) One Phase McGinnis Results	(B) Two Step L.P. Result	(C) Integrated L.P. Results	One Phase McGinnis $100 \times \frac{A-C}{A}$	Two Step L.P. $100 \times \frac{B-C}{B}$
1	116	8234	7409	7108	13.7	4.1
2	117	8151	7193	6895	15.4	4.1
3	88	6143	5583	5317	13.4	4.8
4	103	7338	6754	6655	9.3	1.5
5	87	6123	5380	5146	16.0	4.3
					13.56	3.76

job of fitting the demand curve. There were more hourly periods where demand was met perfectly and a fairly even split between understaffing and overstaffing hours. The McGinnis heuristic assigned 114 of the 116 people to 6 day work weeks and yielded far greater overstaffing and labor costs. The two step L.P. approach assigned only one person to a 6 day work week which resulted in slightly more understaffing. Clearly the two L.P. methods found a better compromise between labor cost and understaffing costs and the integrated L.P. outperformed the two step L.P. relative to both these cost variables.

Using LINDO[16] software on an IBM 3081, the execution time for the suggested

(a)

Start time	M	T	W	Th	F	Sa	Su
6 AM	33	60	57	50	62	--	--
7 AM	--	--	--	--	--	--	--
8 AM	--	--	--	--	--	38	
9 AM	54	--	--	12	20	--	--
10 AM	2	52	54	44	31	--	29

(b)

Shift Pattern	Days-On	Required Patterns
A	M-T-W-Th-F	60
B	T-W-Th-F-Sa	28
C	Th-F-Sa-Su-M	5
D	F-Sa-Su-M-T	1
E	Su-M-T-W-Th-F	19
F	Sa-Su-M-T-W-Th	4

(c)

Iteration	Shifts* Assigned	Number Assigned
1	M_{10}-T_{10}-W_{10}-Th_{10}-Sa_8-Su_{10}	2
2	M_9-T_{10}-W_{10}-Th_{10}-Sa_8-Su_{10}	2
3	T_6-W_6-Th_6-F_6-Sa_8	28
4	M_9-Th_{10}-F_{10}-Sa_8-Su_{10}	5
5	M_9-T_{10}-F_{10}-Sa_8-Su_{10}	1
6	M_9-T_{10}-W_{10}-Th_9-F_{10}-Sa_{10}	12
7	M_9-T_{10}-W_{10}-Th_{10}-F_9	7
8	M_9-T_{10}-W_{10}-Th_{10}-F_9	13
9	M_9-T_{10}-W_{10}-Th_{10}-F_{10}	13
10	M_9-T_{10}-W_{10}-Th_{10}-F_6	1
11	M_6-T_6-W_{10}-Th_6-F_6	1
12	M_6-T_6-W_{10}-Th_6-F_6	3
13	M_6-T_6-W_6-Th_6-F_6	29

(d)

Maximum Start Time Differences	Number of Occurrences
0 hours	29
1 hour	45
2 hours	39
3 hours	0
4 hours	5

Fig. 3. Result of shift assignment heuristic for week-1 data. (a) Required shifts by day and start time. (b) Required shift patterns ignoring start times. (c) Shift assignments using heuristic. (d) Distribution of start time differences.

Table 2. Comparison of three approaches using week-1 data and 116 employees.

	Integrated	McGinnis	Two Step L. P.
1. periods of understaffing	43	25	48
2. periods of overstaffing	23	59	27
3. periods of perfect staffing	18	0	9
4. total understaffing man/hours	773	561	829
5. total overstaffing man/hours	829	1829	749
6. cost of labor	5176	6832	5336
7. cost of understaffing	1932	1402	2073
8. people working 6 day weeks	18	114	1

formulation averaged .12 min. The McGinnis and two-phased L.P. solutions averaged .47 and .32 min of execution respectively on the IBM 3081 computer. The optimization nature of the integrated L.P. method must also account for some of the gain. To demonstrate the shift assignment heuristic, the week-1 solution is illustrated in Fig. 3. The output of the linear program is tabulated in 3(a) and 3(b). The heuristics assignments would occur as they are listed in 3(c). The resulting distribution of start time variations are given in 3(d). In this example the average start time variation was 1.2 hrs with 29 patterns having the same start time for the entire week. Only five patterns had the maximum variation of 4 hrs.

CONCLUSIONS

This paper presented an integrated model of the "days-off" and "shift scheduling" problem. The model was in a decomposable linear programming form. The constraints in the days-off problem such as staff size, consecutive days off requirements, and overtime limits can be reflected in the shift scheduling problem. In addition, constraints in the shift scheduling problem such as hourly demand variations and changes in the cost of understaffing and overstaffing can be reflected in the days-off problem. A heuristic to allocate required shifts to the days-off pattern is presented. The intent of the heuristic is to reduce the occurrence of work patterns with the maximum difference in start times over the week. Compared to an existing heuristic and suggested two step optimization approaches, the integrated model consistently resulted in lower cost solutions. For further reading, the reader may refer to refs. [17–21].

BIBLIOGRAPHY

1. K. R. Baker, Workforce allocation in cyclical scheduling problems: a survey, *Opl Res. Q.*, **27**, 155–167 (1976).
2. B. J. Luce, Employment Assignment Problem. ORSA/TIMS Joint Meeting, Las Vegas, NE (Nov. 1974).
3. B. J. Luce, A Shift Scheduling Algorithm. ORSA 44th National Meeting, San Diego, CA (Nov. 1973).
4. W. B. Henderson & W. L. Berry, Determining optimal shift schedules for telephone traffic exchange operators. *Decision Sci.* **8**, 239–255 (1977).
5. M. Segal, The operator scheduling problem: a network flow approach. *Ops Res.*, **22**, 808–823 (1972).
6. E. G. Keith, Operation Scheduling. *AIIE Trans.*, **11**, 37–41 (1979).
7. C. P. Koelling, *Computer-Aided Personnel Scheduling*. Unpublished Ph.D. Dissertation, Arizona State University, Tempe, AZ (1982).
8. J. M. Field, *Integrated Personnel Scheduling with Flexolift Models*. Unpublished Ph.D. dissertation, Arizona State University Tempe, AZ (1983).
9. R. Tibreweala, D. Philippe & J. Browne, Optimal scheduling of two idle periods. *Management Sci.* **19**, 71–75 (1972).
10. G. Monroe, Scheduling manpower for service operations. *Indus. Engng*, **2**, 10–17 (1970).
11. M. Rothstein, "Scheduling manpower by mathematical programming," *Indus. Engng*, **4**, 29–33 (1972).

12. L. F. McGinnis, W. D. Culver & R. H. Deane, One-and-two-phase heuristics for workforce scheduling. *Comput. & Indus. Engng*, **2**, 7–15 (1978).
13. G. B. Dantzig, A comment on Edie's traffic delays at toll booths. *Ops Res.*, **2**, 339–341 (1954).
14. K. R. Baker, Scheduling a full-time workforce to meet cyclic staffing requirements. *Management Sci.*, **20**, 1561–1568 (1974).
15. J. J. Bartholdi, J. B. Orlin & H. D. Ratliff, Cyclic scheduling via integer programs with circular ones. *Ops Res.*, **28**, 1074–1085 (1980).
16. L. Schrage, *LINDO*. Graduate School of Business, University of Chicago, Chicago, IL (1981).
17. K. R. Baker & M. J. Magazine, Workforce scheduling with cyclic demands and day-off constraints. *Management Sci.*, **24**, 161–167 (1977).
18. K. R. Baker, R. N. Burns & M. Carter, Staff scheduling with day-off and workstretch constraints. *AIIE Trans.*, **11**, 286–292 (1979).
19. K. R. Baker, L. D. Bodin, W. F. Finnegan & R. J. Ponder, Efficient heuristic solutions to an airline crew scheduling problem. *AIIE Trans.*, **11**, 79–85 (1979).
20. J. J. Browne, Simplified scheduling of routine work hours and days off. *Ind. Engng*, **11**, 27–29 (1979).
21. E. S. Buffa, *Modern Production/Operations Management*. John Wiley & Sons, New York, N.Y. (1980).

Solving Large-scale Tour Scheduling Problems

Ahmad I. Z. Jarrah • Jonathan F. Bard • Anura H. deSilva
American Airlines Decision Technologies, P.O. Box 619616, Dallas-Fort Worth Airport, Texas 75261-9616
Graduate Program in Operations Research and Industrial Engineering, Department of Mechanical Engineering, University of Texas, Austin, Texas 78712-1063
Planmatics, Inc., 6315 Poe Road, Bethesda, Maryland 20817

For a given planning horizon, workforce composition and set of labor requirements, personnel scheduling often reduces to solving three problems. The first is concerned with the assignment of days off; the second involves assigning workers to shifts during the day; and the third involves the construction of weekly tours. In many manufacturing facilities, tour scheduling is easy because the start and end times of shifts are invariant, and no work takes place on the weekend. But when daily patterns vary, such as in the airlines, processing, and public service industries, and when part-timers make up a portion of the workforce, the complexity of the overall problem increases dramatically.

This paper presents a new methodology for solving the combined shift and days-off scheduling problem when the labor requirements span less than 24 hours per day. We begin with an integer programming formulation and then introduce a set of aggregate variables and related cuts. When the aggregate variables are fixed the original problem decomposes into seven subproblems (one for each day of the week) that are much easier to solve. A partial enumeration scheme and a heuristic for ensuring feasibility are employed to find upper and lower bounds which converge rapidly to near-optima.

The methodology is applied to tour scheduling at general mail facilities (GMFs). These facilities are located in most urban areas and process millions of mail pieces daily for local and regional distribution. The model accounts for the principal constraints in the U.S. Postal Service labor contract, including half-hour breaks, minimum full-time to part-time ratios, and variable start times. Also considered are four and five day work weeks, and the possibility of assigning workers across labor categories. A full analysis of the Providence, Rhode Island facility is presented.

(*Shift Scheduling; Days-Off Scheduling; Tour Scheduling; Integer Programming; Branch and Bound; U.S. Postal Service*)

1. Introduction

A host of issues surround personnel scheduling, including the assessment of labor requirements, demand forecasting, service level determination, and workforce composition. Once these issues are resolved, up to three interrelated problems must be solved in order to staff a facility. The first deals with the assignment of on and off days to employees (the days-off scheduling problem). The second is concerned with choosing the hours or shifts that an individual will work (the shift scheduling problem). The third considers daily and weekly schedules in an integrated fashion and tries to develop tours over a planning horizon (the tour scheduling problem). If shifts are permitted to span more than a single 24-hour period, the scheduling problem is said to be *continuous*; otherwise it is called *discontinuous*.

0025-1909/94/4009/1124$01.25

For organizations with fixed daily shifts (e.g., a machine shop that operates from 8-to-5, six days a week) the only problem is that of days-off scheduling to insure that each employee works exactly five days a week. For organizations with a fixed 5-day week (e.g., utility offices that close on weekends), the problem is one of scheduling the workforce for various shifts during the day to meet the expected work load. The integrative tour scheduling problem is relevant when management has the flexibility to define shifts and assign days off in a nonuniform manner. Hospital nurses, telephone operators, bus drivers, and airline crews are a few examples of labor groups where such prerogatives exist. In each of these instances, the underlying problem is complicated by the fact that the demand for service fluctuates markedly throughout the day and across the week, and that labor contracts often limit the set of available choices.

The purpose of this paper is to present a comprehensive model of the discontinuous tour scheduling problem and to show how it can be solved efficiently for real world instances. The model is general enough to allow for full and part-time employees with various shift lengths, different days-off patterns, flexible break allocations within specified break windows, 4- and 5-day work weeks, assignments across labor categories, and restrictions on the ratio of full-time to part-time labor hours. In the development, we build on the work of Burns and Carter (1985) who effectively solved the days-off problem, and Bechtold and Jacobs (1990) who devised a clever way of markedly reducing the number of decision variables needed to model shift scheduling. The approach taken is hierarchical. We begin with a relaxed formulation and find optimal solutions to the accompanying integer linear program. Subsequently, we postprocess the output to arrive at high quality, although not necessarily optimal, tours.

In the next section, we present some background material and related research. This is followed in §3 where assumptions and notation are given along with the model. The solution methodology is highlighted in §4. An application concerning personnel scheduling at General Mail Facilities (GMFs) is presented in §5. These facilities typically employ between 200 to 300 clerks and machine operators. Individual machines may be scheduled to run up to 20 hours per day. Results from a full analysis of the Providence, Rhode Island GMF demonstrate the efficiency of the proposed methodology. We are able to solve problems nearly one order of magnitude larger than previously reported, usually within 10 minutes on an IBM 3081-D.

2. Background and Related Research

Depending on the situation, each of the three scheduling problems can be written as an integer linear program of the form:

$$\text{minimize} \quad Z = \sum_{j=1}^{n} c_j x_j \tag{1a}$$

subject to:

$$\sum_{j=1}^{n} a_{jt} x_j \geq r_t, \quad t = 1, 2, \ldots, m \tag{1b}$$

$$x_j \geq 0 \text{ and integer}, \quad j = 1, 2, \ldots, n \tag{1c}$$

where

n respectively, the number of working-day patterns, daily shifts, or weekly tours included in the problem under consideration.

m number of time periods in the planning horizon.

c_j cost of the working-day pattern, daily shift, or weekly tour j.

r_t number of employees required to work in period t.

a_{jt} equals 1 if period t is a work period for working-day pattern, daily shift, or weekly tour j; 0 otherwise.

x_j number of employees assigned to working-day pattern, daily shift, or weekly tour j.

The objective function (1a) minimizes the total cost of the chosen schedule while constraint (1b) ensures that each period t is staffed to meet the requirements r_t. In many applications c_j is taken as one, implying that the working-day patterns, daily shifts, or weekly tours are of equal cost (Baker 1976). Of course, this would not be appropriate when overtime or variable shift lengths are being considered. The above formulation assumes that a set of working-day patterns, daily shifts, or weekly tours are available to select from. While such a set is usually small for the days-off scheduling problem, it can be large for the other two. In such cases, it is common to define working subsets.

2.1. The Days-Off Scheduling Problem

Because of its importance, this problem has been studied extensively over the last 20 years. Baker and Magazine (1977) investigate a seven-days-a-week operation with a fixed weekday demand and a different fixed weekend demand. They further consider four possible modes of operation where employees are entitled to either:

—two days off each week;
—two consecutive days off each week;
—two consecutive days off each week and four days off every two weeks; or
—every other weekend off and two pairs of consecutive days off every two weeks.

Lower bounds are presented for each of the above cases along with a series of algorithms for producing schedules capable of attaining these bounds, thus achieving optimality.

Bartholdi and Ratliff (1978) introduce the concept of "unnetworks," which are problems whose defining matrices are "opposite" to networks in the sense that only two zeros (rather than two ones) appear in each column. They noticed that several days-off scheduling problems have unnetwork incidence matrices because of the fact that employees generally work five days a week and take two days off, and that each working day corresponds to a 1 and each off-day corresponds to a 0 for the a_{jt} elements in problem (1). They exploit this structure by introducing a complementary problem to (1) that seeks to maximize the number of workers that may be given two consecutive days off during the week.

In a related work, Bartholdi et al. (1980) note that several days-off scheduling problems have "row circular" matrices; that is, 0–1 matrices with the ones occurring consecutively, where the first and last row entries are considered consecutive. Two cases that belong to this category are scheduling with two days off each week, and scheduling with two pairs of consecutive days off every two weeks with one of these pairs a weekend. Through a change of variables, it is demonstrated that the matrix of a row circular problem can be transformed into a matrix with only two nonzero entries (either +1 or −1) in each but the last column y_n. If y_n is treated as a parameter the optimal objective function $z(y_n)$ is known to be piecewise convex, and hence a binary search algorithm can be used effectively to bracket the optimal solution.

In a landmark paper, Burns and Carter (1985) consider the following general conditions:

—demand r_t ($t = 1, 2, \ldots, 7$) is variable each day;
—each employee is given at least A out of every B weekends off;
—each employee works 5 out of 7 days; and
—each employee works at most 6 consecutive days;

and formulate three lower bounds. What is most important, though, is that they present a simple procedure that always attains the highest of the three bounds, and hence solves the problem optimally. Emmons and Burns (1991) extended the approach to include hierarchical labor categories, where workers in the higher categories can substitute for those in the lower ones but not vice versa, and where restrictions on the number of employees in each category are stipulated.

Bechtold (1988) addresses a variation of the days-off scheduling problems characterized by multiple locations and multiple objectives. In the model, employees at one location can be transferred to work at other locations on different days of the week. Objectives include minimizing idle time, minimizing the total number of employees, and minimizing the number of transfers during the week. His approach consists of finding the minimal workforce needed to meet the daily aggregate demand across all locations. This value serves as input to a series of models that are used to determine the actual assignments and transfers.

2.2. The Shift Scheduling Problem

This problem is concerned with assigning employees to shifts within a working day. Shifts can have different starting and ending times, different durations particularly when part-timers are allowed, and different break configurations for lunch and resting. This means that the "nice" matrix structures encountered in the days-off scheduling problem are absent. Because of this difficulty, heuristics have been used extensively. The two most common approaches are (1) LP roundoff and (2) decomposition, where some of the constraints are relaxed to induce a special structure that makes the full problem easier to solve. Exact optimization approaches have also been used successfully to solve small to medium sized problems.

Beginning with heuristics, Segal (1974) takes a decomposition approach. He addresses the continuous

scheduling problem of assigning telephone operators to daily shifts to satisfy projected demand. The day is divided into 15-minute periods, and initially the need for breaks is ignored. The resulting problem is modeled and solved as a pure network. The shifts chosen by the model often involve some excess capacity which is used, with the help of a second network model, to provide as many of the required breaks as possible. If at this point some breaks are still unsatisfied, extra employees are added. No information is reported on the quality of the solutions.

Henderson and Berry (1976) look at an application that admits a very large number of shifts. They present two sets of heuristics: the first selects a working subset of shifts from the full range of possibilities, and the second performs the actual scheduling. The selection algorithms include a random shift generator and a maximum difference heuristic which chooses shifts that differ maximally from those shifts already contained in the working subset. The solution heuristics are based on an LP rounding scheme. The authors experiment with various working subset sizes in combination with the heuristics. Good, and often optimal, results are obtained for problems with up to 100 variables.

Bartholdi (1981) considers continuous scheduling problems with column (and hence row) circular matrices containing "intermittent" ones. That is, the ones occur consecutively in each column except for some intermittent zeros. The first and last rows are assumed to be consecutive. These matrices are typical of shift scheduling problems involving continuous shifts interspersed with breaks. In his study, Bartholdi devises an LP round-off heuristic whose accuracy depends on the instance size.

Using a discontinuous model, Mabert (1979) addresses the problem of scheduling check encoders in banks. The total volume of daily check arrivals is assumed to follow a normal distribution from day to day. However, the percentage of the checks arriving within any given hour is assumed to be fixed. As a consequence, model (1) now has a stochastic right hand side, r_t. Chance-constrained programming is used to find solutions.

Morris and Showalter (1983) present a cutting plane algorithm for optimally solving problems with a relatively small number of identical shifts that may overlap from one day to the next. The objective function of the LP relaxation is rounded up to the next higher integer to obtain an initial lower bound. This bound is introduced into the problem and the LP relaxation is solved again. If the new solution is integral, it is also optimal; if not, branch-and-bound is used to see if an integer solution exists with the lower bound as the objective value. If the analysis succeeds, the corresponding solution is optimal; otherwise the lower bound is increased by one, the cut is updated, and the procedure is repeated.

Finally, Bechtold and Jacobs (1990) present a new modeling approach in which shift types rather than shifts are the decision variables. A shift type is identified by a start time, a shift length, and a break window within which the break can start. For the discontinuous case, they demonstrate that an often big reduction in the number of decision variables can be achieved, but at the expense of a new set of constraints needed to insure that each eligible shift type receives a break.

2.3. The Tour Scheduling Problem

Because the days-off and shift scheduling problems are, in effect, special cases of the more general tour scheduling problem, the above discussion applies here as well. Some researchers, though, have taken a holistic approach. Ritzman et al. (1976) study the problem in a post office setting. They use data for a typical week broken down into 84 two-hour periods. Several rules for choosing tours are defined (for example, one rule says add the tour that results in the largest reduction of unprocessed mail). In the algorithm, tours are generated, assigned a fixed weight reflecting the importance of the accompanying rule, and put on a candidate list. A biased random sampling scheme is used to select the next tour from the list.

Mabert and Watts (1982) consider service-oriented industries that limit inventories. Using check encoders at banks as an example, they demonstrate that in this environment the number of weekly tours is quite large when overlapping shifts and part-time employees are allowed. Under certain operating conditions related to daily work stretches and fixed cut-off times for encoding all the checks, working subsets of tours are randomly generated using a biased sampling procedure. The probabilities for this procedure are based upon information gleaned from the solution of daily shift scheduling volumes, as well as daily work loads. The analysis

consisted of a number of experiments aimed at comparing two factors: (1) one of four stretch selection rules, and (2) different sizes for the working subset.

Morris and Showalter (1983) use an LP roundoff heuristic based on working subsets of tours, and report that they were able to obtain a gap of less than one percent between the heuristic and LP solutions for 30 test problems. Love and Hoey (1990) use a two-phase decomposition scheme for a fast-food restaurant application but do not present experimental results. Li et al. (1991) consider employees that differ in productivity, hourly cost, number of available working hours per week, and different days-off constraints. In addition to LP rounding of tour variables, they develop two types of "constraint violation" heuristics that typically start with an infeasible solution and add new tours to achieve feasibility. A dropping heuristic is used to reduce overstaffing. For a study that highlights the benefits associated with nonstandard shift and tour lengths, see Mabert and Showalter (1990).

Loucks and Jacobs (1991) investigate a problem where the workers differ in time of availability and task qualifications. They present a goal programming heuristic that first minimizes overstaffing and then minimizes the deviation between the scheduled and desired hours for each of the employees. Their experimental design consisted of 24 test problems of moderate size. The solutions indicated that anywhere from 30 to 70 workers were needed to satisfy demand. For a comparison of many of the above approaches, see Bechtold et al. (1991).

Bailey (1985) combines the seven daily shift scheduling problems and the days-off selection problem in an integrated formulation which allows for under and overstaffing. His model is similar to ours. The output of the corresponding mixed integer program is postprocessed in an attempt to minimize the maximum difference in shift start times for each of the employees. Results are given for a problem restricted to full-time employees, four feasible start times, 8-hour shifts and 40-hour weeks.

Easton and Rossin (1991) also take an integrated approach and develop a column generation algorithm for constructing working subsets of tours. They begin with a heuristic that finds an initial feasible solution. A basis is then constructed whose dual variables are used in a dynamic program to generate new columns. The dynamic program seeks to compute the reduced cost of each feasible tour in order to find the most attractive entering column for each employee category. Two sets of test problems were examined. The first was characterized by full-time employees working 9-hour shifts, five consecutive days a week with a one-hour break in the middle of the shift. The second allowed for part-time employees working 4-hour shifts but without breaks. In both cases, shifts were required to start at the same time every day. Results for various demand profiles yielded near-optimal solutions in reasonable CPU time.

3. Model Development

In this section, a rather general formulation of the discontinuous tour scheduling problem is presented. The goal is to find the minimum cost pool of employees needed to meet the forecasted labor requirements during each period of the day for each day of the week. We start with a preliminary set of conditions, but later show how to modify the formulation to handle more complex situations.

Assumptions

1. The day is divided into 48 $\frac{1}{2}$-hour periods.
2. Labor requirements are specified for each of the 336 periods in the week.
3. Both full-time and part-time shifts are allowed on any day.
4. A shift must be contained entirely within one day (to accommodate this requirement, a "day" can be conveniently defined as any 24-hour cycle; we use 7:00 am as the starting point).
5. Each full-time employee is entitled to two days off per week (not necessarily consecutive) and is assigned a shift of 8 $\frac{1}{2}$ hours during each of the remaining days.
6. Multiple shift lengths are allowed for each part-time employee.
7. Each shift that exceeds a prescribed duration is entitled to a break of one time period which can be assigned anytime during a prespecified *break window*. The break window consists of a set of contiguous planning periods.
8. No two shift types exist such that the break window of one is a strict subset of the other. (Note, this is equivalent to the absence of *extraordinary overlap* defined by Bechtold and Jacobs.)

9. The ratio of full-time to part-time labor hours should be no less than ρ, a lower limit.

In most operational settings, the classical formulation (1) of the problem resulting from these assumptions requires over a billion variables. As discussed in §2, working subsets of tours are usually generated to reduce this number to a manageable level (Henderson and Berry 1976, Mabert and Watts 1982, Morris and Showalter 1983). The obvious shortcoming of such an approach is that attractive solutions may be overlooked in the reduction process. As an alternative, we develop an approach based on the work of Bechtold and Jacobs on shift scheduling, and Burns and Carter on days-off scheduling that implicitly considers all possible tours by constructing the tours *after* rather than before the problem is solved.

3.1. Implicit Modeling of Breaks

Rather than defining shifts with specified breaks, Bechtold and Jacobs simply consider shift types as variables, where a shift type is identified by a *start time*, a *shift length*, and a *break window* within which the break must begin. This leads to a big reduction in the number of decision variables since a single variable is used to describe all shifts of the same type regardless of when the break begins.

In order to describe the approach, the following notation is needed:

t index for time period

j index for shift type; $j = 1, \ldots, n$

x_j decision variable for the number of employees assigned to shift type j

β_t decision variable for the total number of breaks initiated at time t for all shift types

M set of initial periods, ranked in ascending order, for the break windows associated with all shift types

N set of final periods, ranked in ascending order, for the break windows associated with all shift types

p the earliest period a break can begin for any of the shift types

q the latest period a break can begin for any of the shift types

F_k { j: break window for shift type j lies between time periods p and k }

B_k { j: break window for shift type j lies between time periods k and q }.

Under rather nonrestrictive assumptions paralleling those stated at the beginning of this section, Bechtold and Jacobs show that each shift will receive a break within its window if and only if the following conditions are satisfied:

$$\sum_{t=p}^{k} \beta_t - \sum_{j \in F_k} x_j \geq 0, \quad \forall k \in N \setminus \{q\} \tag{2a}$$

$$\sum_{t=k}^{q} \beta_t - \sum_{j \in B_k} x_j \geq 0, \quad \forall k \in M \setminus \{p\} \tag{2b}$$

$$\sum_{j=1}^{n} x_j - \sum_{t=p}^{q} \beta_t = 0. \tag{2c}$$

Equation (2a) may be thought of as a *forward pass* constraint which ensures that the sum of the breaks initiated during any time span beginning at period p is more than or equal to the total number of shift types with break windows lying completely within the time span. Similarly, (2b) is a *backward pass* constraint which guarantees that the sum of the breaks initiated during any time span ending at period q is more than or equal to the total number of shift types with break windows lying completely within the time span. These two inequalities along with the *balance equation* (2c) ensure that each shift will receive a break within its window. Note that the reduction in the number of decision variables with respect to (1) comes at the expense of the above constraints.

3.2. Exact Lower Bounds for Number of Tours

Burns and Carter derive lower bounds for the number of employees needed to staff a single-shift organization under several rather general days-off restrictions (each employee gets A out of every B weekends off, each employee works 5 days each week, and no employee works more than 6 days in a row). They further show that at least one of the bounds is exact for any instance of the problem. This is done using a constructive proof where a stipulated algorithm is shown to be capable of achieving the highest of the lower bounds.

In this paper we focus on the case where the only days-off restriction is that each full-time employee receives two off-days each week. Hence, the corresponding days-off scheduling problem can be thought of as

a special case of the one addressed by Burns and Carter. Nonetheless, since the constructive algorithm needs to be modified to suit our problem, it is necessary to show that the bounds are still valid.

Consider a days-off scheduling problem with daily requirements r_d ($d = 1, \ldots, 7$), and where each employee works 5 out of 7 days. Two obvious lower bounds on the minimum number of employees needed (ω) are:

$$L1 = \left\lceil \frac{1}{5} \sum_{d=1}^{7} r_d \right\rceil \text{ where } \lceil x \rceil \text{ is the smallest integer}$$

greater than or equal to x, and

$$L2 = \max\{r_d: d = 1, \ldots, 7\}.$$

The first bound $L1$ follows from the fact that each employee works only 5 days (which implies that $5\omega \geq \sum_{d=1}^{7} r_d$). The second bound is self-evident. (Burns and Carter's third lower bound is not applicable here.) Thus we have,

$$\omega = \max[L1, L2] \tag{3}$$

which gives rise to the following algorithm for constructing the days-off assignments once ω is determined from (3):

Step 1: Initialize $S_d = \omega - r_d$; $d = 1, \ldots, 7$.

Step 2: Choose any day k such that $S_k = \max\{S_d: d = 1, \ldots, 7\}$. Put $S_k \leftarrow S_k - 1$.

Step 3: Choose any other day $i \neq k$, such that $S_i \geq 0$. If $S_d = 0$ for all $d \neq k$, set $i = k$. Put $S_i \leftarrow S_i - 1$.

Step 4: Repeat Steps 2 and 3 ω times.

Step 5: If pairs of the form (k, k) are created, swap one element of each such pair with an element of any pair (i, j), where $i \neq k$ and $j \neq k$.

The values of S_d in Step 1 represent the surplus number of employees above the daily requirements and are to be used for assigning off-days. Steps 2 and 3 choose the first and second off-days respectively and decrement the surpluses of the chosen days; at this point nondistinct off-days are allowed. After ω repetitions of Steps 2 and 3, Step 4 resolves nondistinct off-day pairs through swapping.

A simple example is given to demonstrate the algorithm. Consider the case where the labor requirements for days 1 (Monday) through 7 (Sunday) are: 1, 2, 1, 2, 2, 2, 1, respectively. $L1 = \lceil \frac{11}{5} \rceil = 3$ and $L2 = 2$ so the minimum number of employees ω needed is $\max[L1, L2]$, or 3. The initial surpluses for the seven days (Step 1) are:

Day, d	1	2	3	4	5	6	7
Requirement, r_d	1	2	1	2	2	2	1
Surplus, S_d	2	1	2	1	1	1	2

Steps 2 and 3 are applied 3 times. In the first pass the chosen off-days can be (among other possibilities) the pair (1, 7). The surplus vector now becomes:

Surplus S_d	1	1	2	1	1	1	1

Similarly, two other off-day pairs may be chosen as (3, 4) and (6, 7), thus completely determining the weekly days-off schedules for the three employees. It is worth noting that in order to schedule as many weekends off as possible, Step 2 can be preceded by scheduling the weekend off for $\omega - r$ employees and reducing the surpluses for Saturday (day 6) and Sunday (day 7) accordingly, where $r = \min\{r_6, r_7\}$. This will not increase ω. The validity of the algorithm is now shown.

PROPOSITION 1. *The above algorithm generates valid days-off assignments using ω employees, where ω is found from equation* (3).

PROOF. By definition, $\omega \geq r_d$, $d = 1, \ldots, 7$, so all slacks generated in Step 1 are nonnegative. In Steps 2 and 3 the existence of days k and i are guaranteed by lower bound $L1$. That is, $\sum_{d=1}^{7} S_d = 7\omega - \sum_{d=1}^{7} r_d \geq 2\omega$, so the cumulative surplus can provide two days (which may be identical) for each employee. It remains to show that if pairs of the form (k, k) are generated, enough pairs (i, j) exist to permit the swaps in Step 4, where $i \neq k$ and $j \neq k$. Notice that the algorithm starts assigning pairs of the form (k, k) only when $S_d = 0$, all $d \neq k$, $S_k > 0$ and one or more employees must be scheduled. Let e denote the number of remaining unscheduled employees (these employees will be assigned off-days of the form (k, k)), and let c denote the number of scheduled employees that have day k as one of their off-days. From lower bound $L1$, the surplus for day k at this point is at least $2e$; that is,

$S_k = (\omega - r_k) - c \geq 2e$. And since $r_k \geq 0$, we get $\omega - c \geq 2e$, or $\omega - c - e \geq e$.

The left-hand side of the last expression is simply the number of scheduled employees not having k as one of their off-days; i.e., employees with off-days of the form (i, j), where $i \neq k$, and $j \neq k$. Hence, the proof is complete. □

3.3 The Personnel Scheduling Model

The model that we developed for minimizing the cost of staffing a facility over a seven day planning horizon combines (1), (2), and (3) with appropriate modifications and side constraints. The following notation is used in the presentation.

Indices

d index for the day of the week; $d = 1, \ldots, 7$

j index for shift type

k, t index for time period

Parameters

c^f prorated weekly cost of a full-time employee

c_j^p cost of part-time shift type j

f_{jt} 1 if full-time shift type j covers period t; 0 otherwise

p_{jt} 1 if part-time shift type j covers period t; 0 otherwise

p_d earliest period a break can begin for any of the permissible shift types on day d

q_d latest period a break can begin for any of the permissible shift types on day d

γ_j the length of shift type j excluding breaks (hours)

n^f number of full-time shift types

n^p number of part-time shift types

τ_d number of periods considered for scheduling on day d

r_{dt} required number of employees on day d, period t

ρ full-time to part-time labor ratio

Sets

B_{kd}^f { j: break window for full-time shift j on day d lies between k and q_d }

F_{kd}^f { j: break window for full-time shift j on day d lies between p_d and k }

B_{kd}^p { j: break window for part-time shift j on day d lies between k and q_d }

F_{kd}^p { j: break window for part-time shift j on day d lies between p_d and k }

F_d set of all full-time shift types on day d that have breaks

P_d set of all part-time shift types on day d that have breaks

M_d the set of initial periods, ranked in ascending order, for the break windows associated with the shift types of day d

N_d set of final periods, ranked in ascending order, for the break windows associated with the shift types of day d

Decision Variables

β_{dt} total number of breaks initiated on day d in period t for all shift types

x_{dj} number of employees assigned to full-time shift j on day d

y_{dj} number of employees assigned to part-time shift j on day d

ω number of full-time employees needed

Model

$$\text{Minimize } z = c^f\omega + \sum_{j=1}^{n^p} c_j^p \sum_{d=1}^{7} y_{dj} \tag{4a}$$

subject to:

$$\sum_{j=1}^{n^f} f_{jt}x_{dj} + \sum_{j=1}^{n^p} p_{jt}y_{dj} - \beta_{dt} \geq r_{dt}, \tag{4b}$$

$$d = 1, \ldots, 7; \quad t = 1, \ldots, \tau_d$$

$$\sum_{t=p_d}^{k} \beta_{dt} - \sum_{j \in F_{kd}^f} x_{dj} - \sum_{j \in F_{kd}^p} y_{dj} \geq 0, \tag{4c}$$

$$\forall k \in N_d \setminus \{q_d\}; \quad d = 1, \ldots, 7$$

$$\sum_{t=k}^{q_d} \beta_{dt} - \sum_{j \in B_{kd}^f} x_{dj} - \sum_{j \in B_{kd}^p} y_{dj} \geq 0, \tag{4d}$$

$$\forall k \in M_d \setminus \{p_d\}; \quad d = 1, \ldots, 7$$

$$\sum_{j \in F_d} x_{dj} + \sum_{j \in P_d} y_{dj} - \sum_{t=p_d}^{q_d} \beta_{dt} = 0, \quad d = 1, \ldots, 7 \tag{4e}$$

$$40\omega \geq \rho \sum_{j=1}^{n^p} \gamma_j \sum_{d=1}^{7} y_{dj} \tag{4f}$$

$$\omega \geq \frac{1}{5} \sum_{j=1}^{n^f} \sum_{d=1}^{7} x_{dj} \tag{4g}$$

$$\omega \geq \sum_{j=1}^{n^f} x_{dj}, \quad d = 1, \ldots, 7 \tag{4h}$$

x_{dj}, y_{dj}, β_{dt} and ω integer and nonnegative. (4i)

The objective function sums the full-time (first term) and part-time (second term) labor costs. Here we assume that if the cost of a particular part-time shift type varies from day to day (a weekend shift may require time and a half), additional shift types differing only in their hourly costs have been included in the model. However, if full-time shift types with the same length but different hourly costs exist, the formulation would have to be modified slightly (see discussion in §3.5 for enforcing same start times).

Equation (4b) is a set covering type constraint similar to (1b) which stipulates that the sum of the full-time and part-time shifts covering period t on day d, less the number of breaks allocated in period t, should at least match the labor requirement during that period. Equations (4c) through (4e) are the forward, backward, and balance constraints for the seven days of the week used to ensure that each shift requiring a break receives one. Constraint (4f) stipulates that the total full-time hours (left-hand side) should be greater than or equal to ρ, the full-time to part-time minimum ratio, multiplied by the total part-time hours (right-hand side). Finally, (4g) and (4h) place lower bounds on the number of full-time workers, ω, needed to satisfy the various shift requirements during the week.

Solving (4) yields the minimum workforce complement needed to meet the demand; however, it does not specify the tours to be assigned to the full-time employees nor does it enforce any restrictions on the shift start times within each tour. Hence, model (4) is a relaxation of the full problem implying that the task of tour construction must be done in a postprocessing phase. In addition, while *Proposition* 1 assures that ω tours are sufficient to staff the full-time shifts, there is no guarantee that the fluctuation in the start times for each of the tours will be acceptable. This issue is taken up in the computations section where, for the problems investigated, we found that it was always possible to construct valid tours from the solution of (4). When uniform start times are called for, a minor adjustment in the formulation permits us to find individual tours whose start times are the same for each of the five working days.

3.4. Postprocessors

For a simple scheduling problem involving full-time employees only working $8\frac{1}{2}$-hour shifts, 5 out of 7 days a week and a ± 1 hour start window, it can be verified that the new formulation (4) has on the order of 10^3 decision variables, compared to (1) which has order 10^6. The major reason for this difference once again is that no tours are generated in advance. Rather, the approach leaves the task of tour construction until after a solution has been found indicating the number of full-time and part-time shift types needed along with their start times. The concept of implicit modeling of breaks, in turn, serves to further reduce the size of the problem. Nevertheless, in order to explicitly assign breaks to shifts, and to convert the daily shifts into weekly tours for the ω full-time employees, the results of the model must be further processed.

Postprocessor I: Assigning Breaks to Shifts

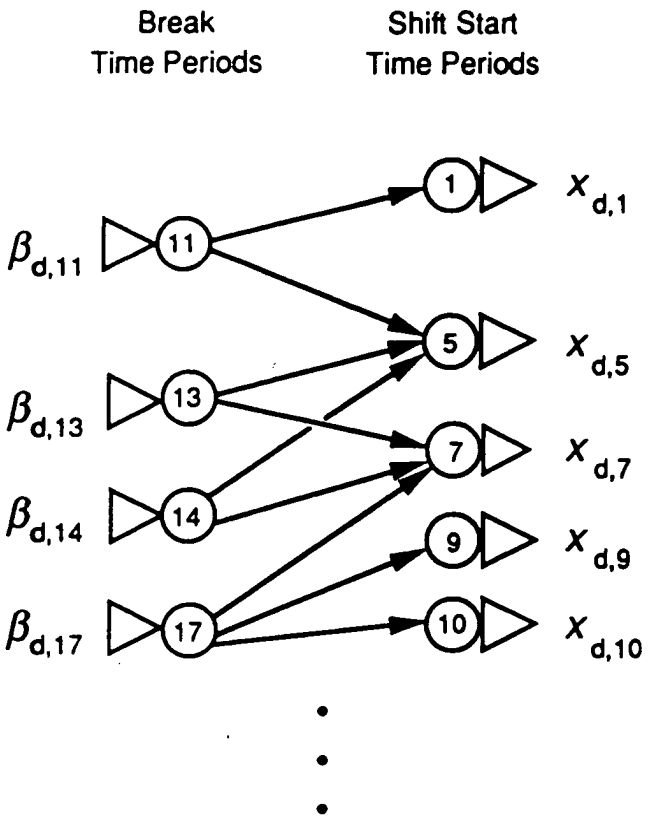
Assigning breaks to shifts is a feasibility problem that can be easily solved using a transportation model for each day of the week, as shown in Figure 1. Here, the nodes on the left represent different time periods during which breaks are initiated, with supplies equal to the number of breaks started at each period of day d. The latter are obtained from the solution of (4). The nodes on the right represent different periods during which shifts are started on day d, with demands equal to the number of shifts started at each period, similarly obtained from the solution of (4). An arc (i, j) appears in this bipartite graph only when shift type j contains period i in its break window. Gaps in the node numbers imply the corresponding variables were found to be zero.

Because we only need a feasible solution any method such as the northwest corner rule will suffice. In order to favor the assignment of breaks near the middle of shifts, one can minimize an objective function whose cost coefficients c_{ij} are given a zero value if period i is at the middle of the break window of shift j, and a proportionally larger value the further period i gets away from the middle.

Postprocessor II: Assigning Shifts to Tours

Although model (4) guarantees that ω full-time workers are sufficient to cover all full-time shifts, it does not provide an explicit assignment of shifts to weekly tours for each of these employees. Accordingly, an additional step is needed. In what follows, it is assumed that the constructed tours should have as low a variation as pos-

Figure 1 Postprocessor I—Partial Network for Assigning Breaks to Shifts on Day d



sible in the start times of their shifts throughout the week. This can be achieved using a simple algorithm that begins by constructing an initial set of tours and then attempts to improve their "quality" by shift swapping. The procedure is:

Step 1. Rank the full-time shifts for each day of the week in ascending order of their start times.

Step 2. Use the days-off algorithm in §3.2 to generate an initial set of ω tours, scheduling for each day the earlier shifts first.

Step 3. Calculate the variance, σ_i^2, for the start times of the shifts of all the tours $i = 1, \ldots, \omega$.

Step 4. Find tour i^* such that: $i^* = \text{argmax}\{\sigma_i^2 : i = 1, \ldots, n\}$.

Step 5. Consider all other tours with at least one shift starting earlier than the latest starting shift in tour i^*, and another shift starting later than the earliest starting shift in tour i^*. If none exists, no improvement is possible for i^*; go to Step 6.

5a. For each such tour j, identify the shift-swapping pattern with tour i^* that will result in the least cumulative variance for the new tours (the best case might be no swapping for all of the shifts).

5b. Identify tour j^* that results in the smallest after-swap cumulative variance among all the tours j. Perform the swap and update the variances of the new tours.

Step 6. Repeat Steps 4 and 5 for a predetermined number of iterations, or as long as variance reduction continues.

A few points merit clarification. First, in an intuitive sense, the likelihood of finding tours with little variation in start times is high due to the fact that labor requirements follow similar patterns from day to day (except perhaps on the weekends). Second, the condition in Step 5 on the earliest and latest times of shifts is necessary for having variance reduction when swapping shifts between tours i^* and j, and hence, can be used to eliminate redundant computations.

A number of possible variants of the above algorithm can be readily conceived. For example, Bailey (1985) attempts to minimize the maximum difference in start times over the week for all tours. As an extension to our basic approach, we adopted a procedure common to tabu search (Glover 1989): rather than terminating the algorithm when tour i^* fails to improve, tour i^* is temporarily placed "on hold" for a certain number of iterations while other tours of lower variance are considered.

Finally, if η denotes the number of iterations after which the procedure is stopped, then the order of complexity of the algorithm is $O(2^7\eta\omega^2)$ or $O(\eta\omega^2)$. To see this, note that in each of the η iterations, ω comparisons are performed in Step 4, up to $\omega - 1$ tours may be considered for swapping in Step 5, and that at most 2^7 swap patterns are possible.

3.5 Extensions to the Model

In building the model, one of our primary concerns was to be able to address staffing issues that may not be a consideration now but may arise in the future. Changing the full-time to part-time labor ratio is one. This can be investigated by simply rerunning the model with different values of ρ in equation (4f). The two other issues that we examine are: (1) the impact of allowing 4-day weeks, and (2) the consequence of enforcing the same

daily start times for the full-time workers. These are slightly more complex and require modifications to the basic model.

Allowing 4-Day Work Weeks

In order to model this situation, let x^1_{dj} be the decision variable associated with the $8\frac{1}{2}$-hour shifts of type j on day d for full-time employees working 5-day weeks, and let x^2_{dj} be the decision variable associated with the $10\frac{1}{2}$-hour shifts of type j on day d for full-time employees working 4-day weeks. Similarly, let n^{f1} and n^{f2} represent the number of full-time shift type for 5-day and 4-day a week full-time employees, respectively. Finally, denote by ω^1 and ω^2 the number of 5-day and 4-day full-time employees. For the new model, the set covering, forward pass, backward, and balance equations may be reproduced as they were in (4), but equations (4f) through (4h) need to be replaced by the following:

$$40(\omega^1 + \omega^2) \geq \rho \sum_{j=1}^{n^p} \gamma_j \sum_{d=1}^{7} y_{dj}$$

$$\omega^1 \geq \frac{1}{5} \sum_{j=1}^{n^{f1}} \sum_{d=1}^{7} x^1_{dj}$$

$$\omega^1 \geq \sum_{j=1}^{n^{f1}} x^1_{dj}, \quad d = 1, \ldots, 7$$

$$\omega^2 \geq \frac{1}{4} \sum_{j=1}^{n^{f2}} \sum_{d=1}^{7} x^2_{dj}$$

$$\omega^2 \geq \sum_{j=1}^{n^{f2}} x^2_{dj}, \quad d = 1, \ldots, 7.$$

The validity of the lower bounds for the 4-day week can be shown in a manner analogous to that of the 5-day week previously discussed. The only remaining step is the replacement of the term $c^f\omega$ in objective function (4a) with $c^f(\omega^1 + \omega^2)$.

Enforcing Same Daily Start Times for Full-Time Workers

Assume that v start time are permitted. Corresponding to each start time τ ($\tau = 1, \ldots, v$) define the variable $x_{d\tau}$ denoting the number of full-time shift types starting at time τ on day d. Also, let $\omega^1, \ldots, \omega^v$ denote the number of full-time employees starting their shifts at times $1, \ldots, v$. Equations (4b) through (4e) remain the same for the new model but with $x_{d\tau}$ replacing x_{dj}, while equations (4f) through (4h) are replaced by:

$$40(\omega^1 + \cdots + \omega^v) \geq \rho \sum_{j=1}^{n^p} \gamma_j \sum_{d=1}^{7} y_{dj}$$

$$\omega^\tau \geq \frac{1}{5} \sum_{d=1}^{7} x_{d\tau}, \quad \tau = 1, \ldots, v$$

$$\omega^\tau \geq x_{d\tau}, \quad d = 1, \ldots, 7; \quad \tau = 1, \ldots, v.$$

In addition, the original term $c^f\omega$ in the objective function is replaced by $c^f(\omega^1 + \cdots + \omega^v)$. If 4-day weeks are to be considered, then additional constraints would have to be added.

The final point to be made here is that if we have shift types that only differ in their cost, the same modeling procedure can be used. This might be the case with evening or weekend shifts, or in situations where it is permissible to assign workers across labor categories; that is, where a skilled worker is assigned to a lower paying job category for one or more shifts but is still paid his higher rate.

4. Solution Approach

Research into solving large scale integer programs over the last few years has shown that good modeling is critical to the solution process. Nevertheless, it has been our experience that realistic instances of problem (4) cannot be tackled with commercial codes in a straightforward manner. (Our attempts to solve (4) directly with OSL (IBM 1991) proved unsuccessful.) In light of this limitation, one of our main goals has been to develop a practical procedure that can be used by industry for both short-term planning and parametric analysis.

4.1. Preliminaries

Inspection of problem (4) reveals that it consists of seven shift scheduling subproblems with complicating side constraints (4f) through (4h); observe that (4f) and (4g) tie the decision variables (x_{dj}, y_{dj}) together over the week. This structure can be exploited by using a decomposition scheme that allows us to solve the smaller shift scheduling subproblems individually, rather than the whole problem at once.

In order to implement this idea, the model has to be modified. Let X^f_d be an integer variable denoting the

total number of full-time shifts on day d. Moreover, consider the set L of part-time shift lengths and let Y^p_{dl} denote the total number of part-time shifts on day d of length $l \in L$ (it is convenient at this point in the presentation to explicitly reference the shift lengths). Finally, let N^p_l denote the set of part-time shift types of length l. The following equations are added to model (4):

$$X^f_d = \sum_{j=1}^{n^f} x_{dj}, \quad d = 1, \ldots, 7 \tag{5a}$$

$$Y^p_{dl} = \sum_{j \in N^p_l} y_{dj}, \quad d = 1, \ldots, 7; \quad l \in L. \tag{5b}$$

Equations (5a) and (5b) can be thought of as surrogate constraints formed by summing the individual shift type variables. Note that fixing the aggregate and workforce variables X^f_d, Y^p_{dl} and ω at some set of (feasible) values $\bar{X}^p_{dl}$, $\bar{Y}^p_{dl}$ and $\bar{\omega}$ decomposes problem (4) into seven independent shift scheduling subproblems with side constraints of the form (5a) and (5b). That is, the objective function (4a) becomes a constant and inequalities (4f)–(4h) become redundant. The corresponding shift scheduling subproblem for day d is to find feasible values of (x_{dj}, y_{dj}) satisfying the following constraints.

Feasibility Subproblem for Day d

$$\sum_{j=1}^{n^f} f_{jt} x_{dj} + \sum_{j=1}^{n^p} p_{jt} y_{dj} - \beta_{dt} \geq r_{dt}, \quad t = 1, \ldots, \tau_d \tag{6a}$$

$$\sum_{t=p_d}^{k} \beta_{dt} - \sum_{j \in F^f_{kd}} x_{dj} - \sum_{j \in F^p_{kd}} y_{dj} \geq 0, \quad \forall k \in N_d \setminus \{q_d\} \tag{6b}$$

$$\sum_{t=k}^{q_d} \beta_{dt} - \sum_{j \in B^f_{kd}} x_{dj} - \sum_{j \in B^p_{kd}} y_{dj} \geq 0, \quad \forall k \in M_d \setminus \{p_d\} \tag{6c}$$

$$\sum_{j \in F_d} x_{dj} + \sum_{j \in P_d} y_{dj} - \sum_{t=p_d}^{q_d} \beta_{dt} = 0 \tag{6d}$$

$$\bar{X}^f_d = \sum_{j=1}^{n^f} x_{dj} \tag{6e}$$

$$\bar{Y}^p_{dl} = \sum_{j \in N^p_l} y_{dj}, \quad l \in L \tag{6f}$$

x_{dj}, y_{dj} integer and nonnegative. (6g)

4.2. Decomposition Algorithm for Augmented Personnel Scheduling Problem

The problem to be solved is model (4) augmented with equations (5a) and (5b). The addition of the latter two constraints does not alter the feasible region. The solution algorithm consists of a truncated branch-and-bound routine and a heuristic which is invoked whenever an all-integer solution fails to materialize within some fixed CPU time. The full methodology is summarized in Figure 2.

(1) (*Master Problem*) Relax the integrality restriction on the variables x_{dj} and y_{dj}, and solve the resultant mixed integer linear program (MILP) requiring only the variables X^f_d, Y^p_{dl} and ω to be integral. Terminate with the optimal solution to the MILP or with a suboptimal solution if a certain predetermined CPU time is exceeded. Denote the solution by $(\bar{X}^f_d, \bar{Y}^p_{dl}, \bar{\omega})$.

(2) (*Subproblems*) Fix the variables X^f_d, Y^p_{dl} and ω at their levels found at Step 1. Solve the resulting seven shift scheduling subproblems (model (6), $d = 1, \ldots, 7$) in (x_{dj}, y_{dj}).

(a) IF the solution, call it $(\bar{x}_{dj}, \bar{y}_{dj})$, is integral for all d and j, STOP with final solution for problem (4);

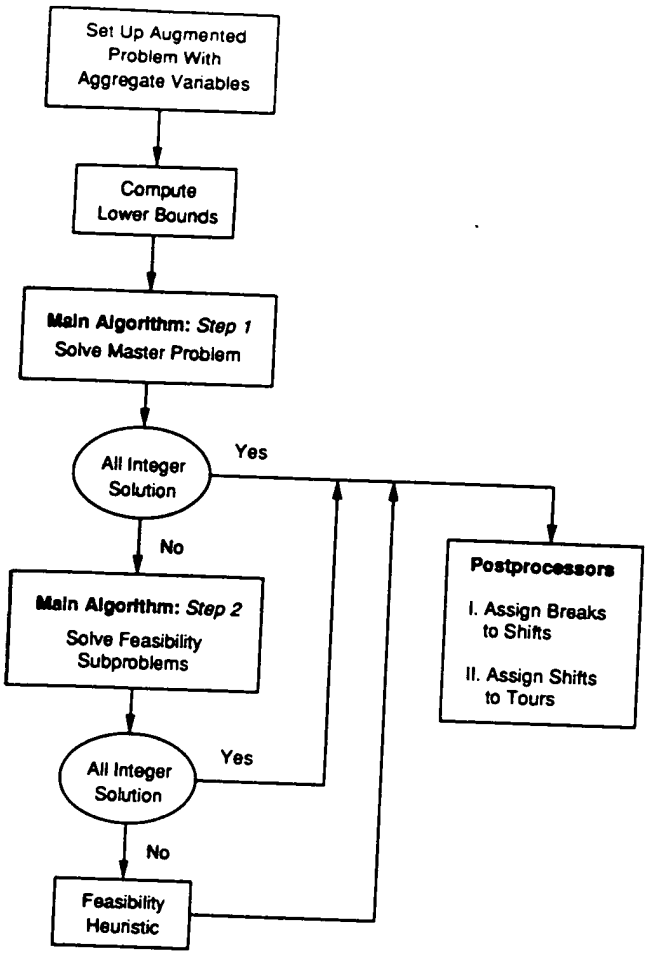
(b) ELSE invoke the heuristic for all infeasible shift scheduling subproblems (6).

The decomposition algorithm fixes the aggregate and workforce variables as branch-and-bound is performed in Step 1. Because the number of integer variables is relatively small, a depth first search always terminates with an integer, although not necessarily optimal, solution to this relaxed problem. Upon completion the corresponding objective function value is $c^f \bar{\omega} + \sum_{d=1,7} \times \sum_{l \in L} c^p_{dl} \bar{Y}^p_{dl}$, where c^p_{dl} represents the cost for a part-time shift of length l on day d. Step 2 attempts to find an all-integer solution for the given values of the aggregate and workforce variables. This leads to the following straightforward observation:

PROPOSITION 2. *If an optimal solution is found for the relaxed problem in Step 1 of the algorithm, and if Step 2 terminates with an all-integer solution prior to invoking the heuristic, then* $(\bar{x}_{dj}, \bar{y}_{dj}, \bar{\omega})$ *is optimal to the personnel scheduling problem* (4).

In general, when using implicit enumeration, the availability of good lower bounds is essential for speeding convergence. An obvious lower bound can be ob-

Figure 2 Solution Methodology



tained from the LP relaxation of (4). A tighter bound requiring little additional work may be derived as follows: define X^f_ω and $Y^p_{\omega l}$ to be the weekly total number of full-time and part-time shifts, and solve a relaxation of the personnel scheduling problem (4) where only these variables along with ω are required to be integer. Of course, if the branch and bound in Step 1 of the algorithm achieves optimality, then a lower bound to (4) is also obtained.

If one or more subproblems are infeasible at the end of Step 2, control passes to the heuristic which attempts to achieve feasibility by sequentially adding shifts. In order to describe its operations, define

$$\text{SLACK} = 40 \sum_{d=1}^{7} \bar{X}^f_d - \rho \sum_{d=1}^{7} \sum_{l \in L} \bar{Y}^p_{dl}$$

as the slack in equation (4f). This value is used to check to see if more part-time shifts can be added without violating the labor ratio restriction.

Feasibility Heuristic

1. (*Add Part-Time Shifts*)

1a) (Store current solution) Set $\hat{Y}^p_{dl} = \bar{Y}^p_{dl}$ for all $l \in L$.

1b) Starting with smallest $l \in L$, cycle through the terms $\bar{Y}^p_{dl}$ in ascending order.

1c) For current l, put $\bar{Y}^p_{dl} \leftarrow \bar{Y}^p_{dl} + 1$

IF for shift length l, SLACK < 0, GOTO Step 2.

ELSE solve the new shift scheduling problem (6).

IF $(\bar{x}_{d:}, \bar{y}_{d\cdot})$ is integral, STOP with the final solution $(\bar{X}^f_d, \bar{Y}^p_{dl}, \bar{\omega}, \bar{x}_{d:}, \bar{y}_{dj})$.

ELSE put $\bar{Y}^p_{dl} \leftarrow \bar{Y}^p_{dl} - 1$.

IF l is the longest part-time shift length, set l to the shortest part-time shift length and

GOTO Step 1d.

ELSE set l to the next longer part-time shift length in the cycle and GOTO Step 1c.

1d) (Add permanent shift and start again) Put $\bar{Y}^p_{dl} \leftarrow \bar{Y}^p_{dl} + 1$, update SLACK and GOTO Step 1b.

2. (*Add New Tour*)

2a) Put $\bar{\omega} \leftarrow \bar{\omega} + 1$, and set $\bar{Y}^p_{dl} = \hat{Y}^p_{dl}$ for all $l \in L$. Update SLACK accordingly.

2b) (Add four additional full-time shifts) For each day $i \neq d$ of the week, identify the set of part-time shifts with the maximum total number of hours that can be reduced due to the addition of an extra full-time shift. From among the six possible days, choose the four with the sets containing the highest number of such hours, breaking ties arbitrarily. Update $\bar{Y}^p_{dl}$ for all $l \in L$ for the chosen four days; update SLACK.

2c) Solve the new shift scheduling problem (6).

IF $(\bar{x}_{d:}, \bar{y}_{d:})$ is integral, STOP with final solution $(\bar{X}^f_d, \bar{Y}^p_{dl}, \bar{\omega}, \bar{x}_{d:}, \bar{y}_{d:})$;

ELSE, GOTO Step 1b.

The heuristic simply loosens those shift scheduling subproblems that failed to reach all-integer solutions in the main algorithm by increasing the total available number of shifts one at a time. The problem is complicated by the requirement of having to maintain the minimum full-time to part-time labor ratio. Step 2 is invoked only when any increase in part-time hours would violate the minimum labor ratio. In this case, the

number of full-time employees, $\bar{\omega}$, is increased by one and the total number of part-time shift types are reset to their original values. Moreover, part-time shifts are eliminated whenever the newly added full-time shifts can replace them.

The final point to be made in this section is that when enforcing the same start times for the full-time workers, the main algorithm needs to be modified to accommodate the additional constraints introduced in §3.5. Fixing the variables $\omega^1, \ldots, \omega^v$ ties the seven shift scheduling problems together precluding the use of the decomposition scheme (i.e., the problem no longer decomposes into seven independent subproblems). Similarly, the heuristic has to be applied to the whole problem at once.

5. Application—Staffing at General Mail Facilities

In most major population centers, mail collected locally and mail arriving from outside the area is centrally processed at a general mail facility (GMF). Each of the approximately 280 GMFs in the U.S. is configured with a host of mechanized and automated equipment designed to sort, sequence, and distribute the various classes of mail. Processing operations on each machine span up to 20 hours per day, seven days a week, with much of the remaining time put aside for preventive maintenance.

The models and algorithms presented in this paper have been developed with the GMF personnel scheduling problem in mind. The equipment used for processing can be classified into four groups based on the required skill level and operator training involved. The four groups are:

—Group 1: Bar Code Sorters (BCS), Multi-Line Optical Character Readers (MLOCR), Remote Bar Code Sorters (RBCS), and Sequencing Bar Code Sorters (SEQ.BCS).
—Group 2: Remote Video Encoding Systems (RVES) and Multi-Position Letter Sorting Machines (MPLSM).
—Group 3: Advanced Face Canceller Systems (AFCS) and Bundle Sorters (BUNDS).
—Group 4: Manual operations.

Because of union agreements, the personnel scheduling problem naturally decomposes into four independent problems by labor category. In each category, we are interested in specifying both the daily shifts and the weekly days-off for each employee. Daily staffing needs do not span the usual 8 to 5 pattern; rather, personnel may be required to run the various machines throughout most of the day and evening in order to get the mail processed and ready for delivery the following morning. Both full-time regular (FTR) employees and part-time flexible (PTF) employees must be considered since the current labor contract at the U.S. Postal Service (USPS) allows up to 10% of the workforce to be part-time.

Flexibility exists in scheduling FTRs with respect to their start-times and in the positioning of their lunch breaks; the rule being that somewhere between the fourth and the sixth working hour a half-hour break must be provided. In assigning shifts, it is desirable that the same shift, or at least one with approximately the same start time (within ±1 hour) be assigned each of the working days. Regarding days-off scheduling, each FTR receives two, preferably consecutive, days off each week. A goal is to have at least one of these two days fall on a weekend.

A PTF can be scheduled at any time during the week for up to a total of 39 working hours. Once called, though, he or she cannot be assigned to less than four hours a day, and if called for six or more hours must be granted a half-hour lunch break somewhere between the fourth and sixth working hour. Consequently, part-time scheduling involves the juggling of a large number of shifts.

5.1. Input Data

The Providence GMF served as the testbed for the analysis. Input data consisted of the number of employees needed during each half-hour period for each of the four machine groups. These values were derived from a series of equipment selection and machine scheduling models developed in companion works (see Jarrah et al. 1992a, 1992b) and are available from the authors. The labor cost for FTRs is assumed to be \$12/hr for machine groups 1 and 3, and \$14/hr for groups 2 and 4. The labor costs for PTFs are \$9/hr for groups 1 and 3, and \$12/hr for groups 2 and 4. In the baseline scenario the FTRs work 5-day weeks, $8\frac{1}{2}$ hours per day, and are entitled to a half-hour lunch break.

Five shift lengths are permitted for PTFs: 4, 5, $6\frac{1}{2}$, $7\frac{1}{2}$ and $8\frac{1}{2}$ hours; only the last three are eligible for a half-

hour break. Also, it is desirable for the start time of the daily shifts for each FTR not to fluctuate by more than ±1 hour throughout the week. In a later section, the effect of allowing 4-day weeks and enforcing same daily start times for the FTRs is investigated.

5.2. Computational Results

Twenty-eight runs were made to test the algorithms. Table 1 gives the problem size and input parameters for each of the runs. The four blocks of data are for the four groups: BCS-MLOCR-RBCS-SEQ.BCS (runs 1–7), RVES-MPLSM (runs 8–14), AFCS-BUNDS (runs 15–21), and MANUAL (runs 22–28). As can be seen, the problems vary in size by a factor of two with run 8 being the smallest in both number of variables and constraints, run 27 the largest in number of variables, and run 28 the largest in number of constraints. The average subproblem size shows some variability as well, with

Table 1 Problem Characteristics

Problem No.	No. of Var. (Problem)	No. of Const. (Problem)	Avg. No. of Var./Const. (Subproblems)	Description
Group 1:				
1	1,345	702	198/99	90-10 rule, 5-day weeks, flexible start-times
2	1,345	702	198/99	85-15 rule, 5-day weeks, flexible start-times
3	1,345	702	198/99	80-20 rule, 5-day weeks, flexible start-times
4	1,345	702	198/99	75-25 rule, 5-day weeks, flexible start-times
5	1,345	702	198/99	70-30 rule, 5-day weeks, flexible start-times
6	1,479	718	218/100	90-10 rule, 5 & 4-day weeks, flexible start-times
7	1,367	878	–	90-10 rule, 5-day weeks, same start-times
Group 2:				
8	806	471	121/66	90-10 rule, 5-day weeks, flexible start-times
9	806	471	121/66	85-15 rule, 5-day weeks, flexible start-times
10	806	471	121/66	80-20 rule, 5-day weeks, flexible start-times
11	806	471	121/66	75-25 rule, 5-day weeks, flexible start-times
12	806	471	121/66	70-30 rule, 5-day weeks, flexible start-times
13	863	487	135/67	90-10 rule, 5 & 4-day weeks, flexible start-times
14	819	562	–	90-10 rule, 5-day weeks, same start-times
Group 3:				
15	908	498	157/82	90-10 rule, 5-day weeks, flexible start-times
16	908	498	157/82	85-15 rule, 5-day weeks, flexible start-times
17	908	498	157/82	80-20 rule, 5-day weeks, flexible start-times
18	908	498	157/82	75-25 rule, 5-day weeks, flexible start-times
19	908	498	157/82	70-30 rule, 5-day weeks, flexible start-times
20	988	512	171/83	90-10 rule, 5 & 4-day weeks, flexible start-times
21	926	614	–	90-10 rule, 5-day weeks, same start-times
Group 4:				
22	1,457	750	214/106	90-10 rule, 5-day weeks, flexible start-times
23	1,457	750	214/106	85-15 rule, 5-day weeks, flexible start-times
24	1,457	750	214/106	80-20 rule, 5-day weeks, flexible start-times
25	1,457	750	214/106	75-25 rule, 5-day weeks, flexible start-times
26	1,457	750	214/106	70-30 rule, 5-day weeks, flexible start-times
27	1,607	766	236/107	90-10 rule, 5 & 4-day weeks, flexible start-times
28	1,483	947	–	90-10 rule, 5-day weeks, same start-times

the smaller problems appearing in the RVES-MPLSM group and the larger ones in the MANUAL group. Recall that in runs 7, 14, 21 and 28, the problems are not decomposable, and hence, the entries for average subproblem size are left blank. The variability in problem size across groups is due to the different number of scheduling periods for each of them, while the variability within the groups is due to changes in labor rules.

All algorithms were written in FORTRAN and run on the IBM 3081-D at the University of Texas. OSL's simplex based codes were used as solvers. The MPS input files for OSL were generated with the algebraic modeling language XML (Marsten 1981). XML in turn requires LINDO-like input files which were created using a uniquely tailored FORTRAN code. Storage requirements for the MPS files ranged from 1 to 2.5 megabytes.

The branch-and-bound portions of the algorithms use OSL's default subroutine; however, initial branching is stipulated to follow the order in which the variables appear in Step 1 (master problem) of the main algorithm, and a depth-first search is enforced until a feasible solution is found. Subsequently, OSL is left to follow its own branching and node selection rules. This scheme proved instrumental in locating a good feasible solution early on. The branch-and-bound is stopped if the optimal solution to the master problem is not reached within 300 CPU seconds after the LP relaxation is solved, except in runs 7, 14, 21 and 28 where the cutoff time is increased to 500 CPU seconds due to the larger number of integer variables involved. Finally, Postprocessor II (assigning shifts to tours) is stopped whenever the number of iterations equals the number of employees being scheduled, or if 12 iterations elapse without any reduction in the variance of the start times.

Tables 2 and 3 list the solutions and CPU times for the 28 runs, respectively. Column (2) displays the lower bounds obtained in each case. These bounds were derived in one of two ways: (i) from the LP relaxation of problem (4); or (ii) from the relaxation involving the weekly aggregate variables as described in §4.2 (a 300 CPU sec limit was placed on these computations as well). Column (3) gives the value of the best feasible solution uncovered. The percent gap between the upper and lower bounds is indicated in column (4), and is seen to be consistently less than 1% except for runs 20, 27 and 28. In particular, the gap was zero (to the second decimal place) in 14 instances and less than 0.2% in nine others.

Columns (5) through (12) report the number of 5-day FTRs, 4-day FTRs, 4-hour PTF shifts, 5-hour PTF shifts, $6\frac{1}{2}$-hour PTF shifts, $7\frac{1}{2}$-hour PTF shifts, and $8\frac{1}{2}$-hour PTF shifts, respectively. As expected, the shorter PTF shifts are used more often than the longer ones. Column (13) lists the slack, or difference between assigned labor hours and required labor hours; the variability in this entry will be discussed shortly.

In Table 3, columns (14) and (15) give the solution times for the LP relaxation and lower bound results, respectively. Column (16) gives the CPU time for the branch-and-bound (BB1) computations in Step 1 of the main algorithm. Note that in 2 cases the optimal solution was found within the 300 or 500 CPU seconds allotted for this phase. This, coupled with the 12 other cases where the gap was zero, is strong testimony to the effectiveness of the aggregate variable cuts (eqs. 5a and 5b). Column (17) reports the number of nodes evaluated during branch-and-bound in Step 1, and column (18) gives the node at which the best solution was found.

Column (19) reports CPU times for the branch-and-bound (BB2) in Step 2 of the main algorithm. Recall that we are attempting to find feasible solutions to the subproblems at this stage. Note that for runs 13 and 21, these computations were not needed; i.e., fixing the aggregate and workforce variables at integer values yielded a solution in which all the shift variables (x, y) were also integral. This was not the case for the other runs; column (20) gives the number of subproblems that had to be solved in each instance. This number was kept to a minimum by recognizing days with identical staffing requirements and aggregate variable values (obtained in Step 1 of the algorithm), and solving only one subproblem for such days. Problems 7, 14, 21 and 28 do not decompose into independent shift scheduling problems as discussed earlier, and hence, the branch-and-bound in Step 2 was applied to the whole problem. A corresponding increase in computational burden was experienced.

It is rather remarkable that in all the runs, all-integer solutions were obtained at relatively low computational cost. Hence, the heuristic (column 21) was never in-

Table 2 Results for the Four Machine Groups—Cost and Workforce

Problem No. (1)	Lower Bound ($)/wk (2)	Upper Bound ($)/wk (3)	(3)-(2)/(2) (%) (4)	ω (5)	$F1$ (6)	$F2$ (7)	$P1$ (8)	$P2$ (9)	$P3$ (10)	$P4$ (11)	$P5$ (12)	Slack (hr) (13)
Group 1:												
1	52,965	52,983	0.02	102	510	-	83	23	-	-	-	411
2	51,450	51,522	0.14	95	475	-	67	54	20	-	-	342
3	49,978	50,052	0.15	88	440	-	64	60	52	-	-	272
4	48,508	48,519	0.03	81	405	-	57	81	73	-	-	195
5	47,248	47,313	0.14	75	375	-	35	125	82	-	-	141
6	49,806	49,878	0.14	96	290	152	16	2	58	-	-	146
7	52,965	52,974	0.02	102	510	-	84	22	-	-	-	410
Group 2:												
8	109,380	109,380	0.00	180	900	-	195	-	-	-	-	1,496
9	103,260	103,260	0.00	162	810	-	285	-	-	-	-	1,136
10	97,820	97,820	0.00	146	730	-	365	-	-	-	-	816
11	93,818	93,818	0.00	133	665	-	392	38	-	-	-	594
12	90,398	90,107	0.00	121	605	-	392	98	-	-	-	414
13	89,992	90,107	0.13	148	285	364	115	-	28	3	1	93
14	110,808	110,808	0.00	182	910	-	202	-	-	-	-	1,604
Group 3:												
15	9,804	9,804	0.00	19	95	-	19	-	-	-	-	104
16	9,204	9,204	0.00	17	85	-	29	-	-	-	-	64
17	8,967	8,967	0.00	16	80	-	30	-	3	-	-	37
18	8,757	8,757	0.00	15	75	-	30	-	8	-	-	36
19	8,547	8,547	0.00	14	70	-	30	-	13	-	-	26
20	9,227	9,324	1.07	18	65	20	9	3	2	-	-	51
21	9,804	9,813	0.06	19	95	-	18	1	-	-	-	106
Group 4:												
22	34,560	34,560	0.00	57	285	-	60	-	-	-	-	331
23	32,860	32,860	0.00	52	260	-	85	-	-	-	-	231
24	30,820	30,820	0.00	46	230	-	115	-	-	-	-	111
25	29,482	29,680	0.67	42	210	-	129	-	1	2	3	48
26	28,765	28,979	0.75	39	195	-	129	11	1	-	9	20
27	30,545	30,914	1.20	51	50	164	39	8	3	-	-	65
28	35,918	36,504	1.60	60	295	-	66	-	-	-	-	475

voked. These findings are consistent with those of Morris and Showalter (1983) for single-day, single-shift problems.

Columns (22) and (23) in Table 3 report the CPU times for the two postprocessors. While the first of these solved quickly, some effort was needed for the second, except for problems 7, 14, 21 and 28, where all tours are explicitly enforced to start at the same time each working day. The disproportionate amount of time spent on assigning shifts to tours is due to the large number of swaps that were attempted to improve the quality of the final schedules. In the worst case, the running time of exchange heuristics are often exponentially bounded. We will return to this point presently. Finally, column (24) gives the total CPU seconds for each of the problems. Solution times range between

Table 3 Results for the Four Machine Groups—CPU Time†

Problem No. (1)	LP CPU (sec) (14)	LB CPU (sec) (15)	BB1 CPU (sec) (16)	No of Nodes (17)	Best Node (18)	BB2 CPU (sec) (19)	No. of Sub. (20)	Heur CPU (sec) (21)	PI CPU (sec) (22)	PII CPU (sec) (23)	Total CPU (sec) (24)
Group 1:											
1	64.4	57.8	300.0	223	103	21.1	5	-	0.3	25.6	469.2
2	64.0	289.5	300.0	317	82	25.8	6	-	0.3	24.7	704.2
3	64.9	300.0	300.0	224	37	39.6	6	-	0.3	38.1	742.9
4	61.0	300.0	300.0	280	22	28.4	6	-	0.3	25.8	715.5
5	74.6	300.0	300.0	246	155	26.3	4	-	0.3	22.8	724.0
6	93.2	172.7	300.0	163	103	10.7	4	-	0.3	17.0	593.9
7	160.0	155.0	500.0	121	85	195.0	-	-	0.3	0.3	1,010.6
Group 2:											
8	20.4	1.5	7.6	14	14	4.0	4	-	0.2	13.2	46.9
9	20.5	1.1	8.0	15	15	1.0	4	-	0.2	9.2	40.0
10	21.2	0.0	2.0	3	3	2.6	3	-	0.2	1.4	27.4
11	22.0	1.2	3.5	6	6	3.7	4	-	0.2	8.3	38.9
12	26.5	1.3	6.5	10	10	4.9	3	-	0.2	15.0	54.4
13	33.9	300.0	300.0	448	423	-	-	-	0.3	8.0	642.2
14	34.6	1.1	8.4	12	12	8.7	-	-	0.2	0.5	53.5
Group 3:											
15	24.4	2.6	11.7	13	13	5.7	3	-	0.2	0.7	45.3
16	31.5	2.2	12.5	10	10	3.0	2	-	0.2	0.3	49.7
17	27.8	17.1	14.9	11	11	2.83	2	-	0.2	0.6	63.4
18	33.6	22.6	19.3	25	25	3.2	2	-	0.2	0.9	79.8
19	27.9	21.3	26.1	34	34	4.4	3	-	0.2	0.9	80.8
20	39.6	149.0	300.0	327	182	23.7	4	-	0.2	2.7	515.2
21	58.5	5.6	500.0	202	90	-	-	-	0.2	0.1	564.4
Group 4:											
22	54.6	3.1	40.6	23	23	16.3	5	-	0.2	26.1	140.9
23	65.9	9.1	40.0	23	23	19.3	4	-	0.2	11.8	146.3
24	64.1	0.0	16.5	9	9	19.0	6	-	0.2	9.8	109.6
25	83.5	10.5	300.0	141	63	34.0	7	-	0.2	12.9	441.1
26	102.5	300.0	300.0	166	161	21.4	3	-	0.2	12.1	736.2
27	131.8	300.0	300.0	184	104	42.3	3	-	0.2	8.5	742.8
28	209.5	300.0	500.0	111	89	214.1	-	-	0.2	0.2	1,224.0

† All computations performed on an IBM 3081-D.

27.4 CPU seconds (run 10) and 20.4 CPU minutes (run 28).

Four- and Five-Day Work Weeks

For each of the machine groups, a substantial cost reduction is achieved when 4-day weeks are allowed (5.8%, 17.6%, 4.9%, and 10.5%, respectively). To illustrate, for group 1, a combination of fifty-eight 5-day-a-week and thirty-eight 4-day-a-week FTRs along with sixteen 4-hour, two 5-hour, and fifty-eight 6-hour PTF shifts obtained in run 6 were able to match the staffing requirements more closely than the 102 5-day-a-week FTRs with eighty-three 4-hour and seven 5-hour shifts obtained in run 1. This is evident in the reduction of slack hours from 411 to 146. It can be verified that this simple modification translates into annual savings in

excess of 1.3 million dollars for the whole facility (ignoring the time value of money within the year).

Enforcing Same Start Times

When each FTR is required to start his or her shift at the same time each day, the percent increases in staffing costs above the baseline were 0.0%, 1.3%, 0.1%, and 5.7% for the four groups, respectively. It is curious to note that for group 1, the cost actually went down by $9 from run 1 to run 7. This can be explained by examining columns (8) and (9) in Table 2. As can be seen, one additional part-time hour is contained in the solution for run 1, implying suboptimality.

Somewhat surprisingly, the results for the baseline runs and this scenario were quite similar. A partial explanation owes to the existence of extensive slack in both cases (column 13). The requirement that work assignments be continuous during the day precludes the possibility of redistributing the slack and eliminating tours. An implication for management is that it may be desirable to implement a same start time policy and incur the extra cost (a total of $3,381 per week) in exchange for a likely improvement in employee morale. Most workers prefer the same shift every day.

Changing the Full-Time to Part-Time Labor Ratio

The current USPS labor contract requires that the ratio of full-time to part-time hours be no less than 0.9. The question arises as to the benefits that can be gained by reducing this number. From management's point of view, if the cost-savings are expected to be minimal, then it might not be worth confronting the union on this issue. If the potential savings are significant, however, a tough negotiating stance may be justified.

The effect of changing the minimum full-time to part-time ratio from 90-10 down to 70-30 in increments of 5% can be seen in the output of runs 1–5, 8–12, 15–19, and 22–26. For example, for group 1 it is observed that a large reduction in cost occurs at each 5% increment with a total of $5,670 per week (or 10.7%) between the 90-10 and 70-30 rules. The same results are realized for the other machine groups. The corresponding reduction in slack with each 5% increment in the labor rule reveals that the savings stem not only from the lower cost of part-time workers, but also from the increased ability to match labor requirements with shift lengths. The larger pool of four and five hour part-time shifts makes this possible.

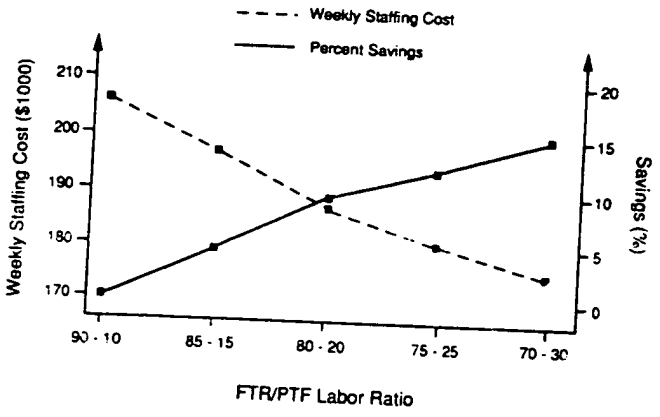
Figure 3 depicts the parametric relationship between aggregate cost and the labor ratio for the entire facility. Notice that the incremental savings per 5% change in labor rule is less for the 80-20 to 70-30 interval than for the 90-10 to 80-20 interval. The reason is that slack reduction tends to become more difficult after a certain amount of part-time shifts are made available. Nevertheless, the findings point up the desirability of negotiating changes in the labor ratio since switching to the 70-10 rule can yield total annual savings in excess of $1.9 million.

Assessment of Postprocessor II

It is important to investigate the quality of the derived schedules with respect to the variation in the daily start times for the FTRs. Table 4 gives the number of FTRs for each of the 28 test problems, along with statistics regarding the tours before and after Postprocessor II is applied. These statistics consist of the percent of tours having their start times within a two-hour band, the worst (widest) band for any of the tours, and the percent of employees assigned weekends off or two consecutive days off (other than weekends).

As can be seen, in quite a few cases the postprocessor is successful in narrowing the spread in employee start times; however, this sometimes occurs at the expense of reducing the number of weekends and consecutive days off. To illustrate, for run 1, while the widest start time variation was reduced from 11 to 3.5 hours, the percentages of weekends and consecutive days off dropped from 21.6% and 44.1% to 19.6% and 32.3%, respectively. In those instances where some tours have unreasonable variation in start times (e.g., some of the

Figure 3 **Affect of Changing FTR to PTF Labor Ratio**



tours associated with runs 26 and 27) more part-time or full-time shifts are necessary to reduce the spread.

6. Summary and Conclusions

Labor costs still dominate many organizations. In an increasingly competitive environment, good staff schedules can make the difference between mere survival and a profitable operation. The personnel scheduling problems solved in this paper are the largest and most complex addressed in the literature to date and are representative of those being solved continually in a wide range of industries. Near-optimal solutions were obtained in reasonable CPU times for problems with up to 1,600 integer variables.

From a methodological point of view, an interesting extension to this work would be to allow different days-off patterns for the full-time employees. This can be done by appropriately modifying constraints (4g) and (4h) (as was done in §3.5 to accommodate 4-day work

Table 4 Quality of Schedules Before and After Post Processing

Problem No.	% Within 2-hr Band	Worst Band (hrs)	% Weekend Off	% Consec. Days Off	% Within 2-hr Band	Worst Band (hrs)	% Weekend Off	% Consec. Days Off
Group 1:								
1	65.7	11.0	21.6	44.1	72.6	3.5	19.6	32.3
2	48.4	11.0	27.4	52.7	50.0	5.5	28.4	44.2
3	32.9	11.0	6.8	52.2	45.4	5.5	6.8	36.4
4	35.8	11.0	9.9	58.0	71.6	3.5	9.9	40.7
5	38.7	11.0	5.3	37.3	53.3	3.5	8.9	28.0
6	36.9	11.0	34.4	46.9	61.5	5.5	32.3	43.8
7	100.0	0	27.2	37.9	100.0	0	27.2	37.9
Group 2:								
8	78.3	4.5	18.9	66.2	84.4	3.5	17.2	50.5
9	83.9	4.5	18.5	56.2	88.9	3.0	21.6	44.4
10	84.9	4.5	16.4	72.6	84.9	4.5	16.4	72.6
11	82.0	3.5	13.5	72.9	85.7	3.0	15.8	64.0
12	76.9	3.5	11.6	71.1	77.7	3.5	10.7	59.5
13	76.4	5.5	10.8	60.8	89.2	5.5	9.5	54.1
14	100.0	0	30.7	36.8	100.0	0	30.7	36.8
Group 3:								
15	100.0	2.0	31.6	15.8	100.0	2.0	31.6	15.8
16	88.3	2.5	25.0	17.6	88.3	2.5	35.0	17.6
17	87.5	5.5	31.2	12.5	100.0	2.5	31.2	12.5
18	86.7	6.5	26.7	6.7	93.4	2.5	26.7	6.7
19	71.4	7.0	21.4	21.4	92.8	2.5	21.4	21.4
20	55.5	8.0	22.2	33.3	66.6	7.0	22.2	33.3
21	100.0	0	31.5	15.8	100.0	0	31.5	15.8
Group 4:								
22	33.3	13.0	8.8	38.6	45.6	6.5	8.8	24.6
23	34.6	13.0	7.6	30.8	42.3	5.5	5.8	15.4
24	32.6	13.0	21.7	32.6	30.4	5.0	21.7	23.9
25	26.2	13.0	19.9	38.1	52.4	5.0	16.7	21.4
26	35.9	13.0	15.4	30.8	61.6	8.5	15.4	10.3
27	64.7	12.0	29.4	60.8	84.3	12.0	27.4	56.9
28	100.0	0	45	16.7	100.0	0	45	16.7

weeks), and/or by introducing additional constraints on ω to reflect the new days-off restrictions (see Burns and Carter 1985). Another extension would be to develop a technique for enforcing bands on the start times of full-time employees. It is easy to enforce same start times but not bands. We now rely on a postprocessor to assign shifts to tours; however, the accompanying heuristic offers no assurance that each shift will start within a given band. Finally, the biggest challenge for future research is extending the methodology to solve the continuous tour scheduling problem.

Regarding the main algorithm, we have not yet been able to fully explain its efficiency. The feasibility subproblem (6) at Step 2, containing on average over 1000 integer variables, always terminated with an integer solution within a minute, and sometimes within seconds except for those runs where all seven subproblems have to be solved simultaneously. But even there, the solution times for BB2 were only a few minutes, and for run 21 the subproblems did not have to be solved at all. One conjecture for explaining these results is that the surrogate cuts (5) define a facet of the original problem's convex hull, thereby increasing the likelihood that a solution to the LP relaxation is integral. More work needs to be done, though, to substantiate this.[1]

[1] A portion of this work was supported by a grant from the Texas Higher Education Coordinating Board's Advanced Research and Technology Programs.

References

Bailey, J., "Integrated Days Off and Shift Personnel Scheduling," *Computers & Industrial Engineering*, 9, 4 (1985), 395–404.

Baker, K. R., "Workforce Allocation in Cyclical Scheduling Problems: A Survey," *Oper. Res.*, 22, 1 (1976), 155–167.

—— and M. Magazine, "Workforce Scheduling with Cyclic Demands and Day-off Constraints," *Management Sci.*, 24, 2 (1977), 161–167.

Bartholdi, J. J., III and H. D. Ratliff, "Unnetworks with Applications to Idle Time Scheduling," *Management Sci.*, 24, 8 (1978), 850–858.

——, J. B. Orlin and H. D. Ratliff, "Cyclic Scheduling via Integer Programs with Circular Ones," *Operations Res.*, 28, 5 (1980), 1074–1085.

——, "A Guaranteed-Accuracy Round-off Algorithm for Cyclic Scheduling and Set Covering," *Oper. Res.*, 29, 3 (1981), 501–510.

Bechtold, S. E., "Implicit Optimal and Heuristic Labor Staffing in A Multi-Objective Multi-Location Environment," *Decision Sciences*, 19, 2 (1988), 353–372.

—— and L. W. Jacobs, "Implicit Modeling of Flexible Break Assignments in Optimal Shift Scheduling," *Management Sci.*, 36, 11 (1990), 1339–1351.

——, M. J. Brusco and M. J. Showalter, "A Comparative Evaluation of Labor Tour Scheduling Methods," *Decision Sciences*, 22, 4 (1991), 683–699.

Burns, R. N. and M. W. Carter, "Work Force Size and Single Shift Schedules with Variable Demands," *Management Sci.*, 31, 5 (1985), 599–607.

Easton, F. F. and D. F. Rossin, "Sufficient Working Subsets for the Tour Scheduling Problem," *Management Sci.*, 37, 11 (1991), 1441–1451.

Emmons, H. and R. N. Burns, "Off-Day Scheduling with Hierarchical Worker Categories," *Oper. Res.*, 39, 3 (1991), 484–495.

Glover, F., "Tabu Search, Part I," *ORSA J. Computing*, 1, 3 (1989), 190–206.

Henderson, W. and W. Berry, "Heuristic Methods for Telephone Operator Shift Scheduling: An Experimental Analysis," *Management Sci.*, 22, 12 (1976), 1372–1380.

IBM, *Optimization Subroutine Library: Guide and Reference*, International Business Machines Corporation, Dept. 52QA/MS 511, Kingston, NY, 1990.

Jarrah, A. I. Z., J. F. Bard and A. H. deSilva, "A Heuristic For Machine Scheduling at General Mail Facilities," *European J. Oper. Res.*, 63 (1992), 192–206.

——, —— and ——, "Equipment Selection and Machine Scheduling at General Mail Facilities," *Management Sci.*, 40, 8 (1994).

Li, C., E. P. Robinson and V. A. Mabert, "An Evaluation of Tour Scheduling Heuristics with Differences in Employee Productivity and Cost," *Decision Sciences*, 22, 4 (1991), 700–718.

Loucks, J. S. and F. R. Jacobs, "Tour Scheduling and Task Assignment of a Heterogeneous Work Force: A Heuristic Approach," *Decision Sciences*, 22, 4 (1991), 719–739.

Love, R. and J. R. Hoey, "Management Science Improves Fast-Food Operations," *Interfaces*, 20, 2 (1990), 21–29.

Mabert, V. A., "A Case Study of Encoder Shift Scheduling Under Uncertainty," *Management Sci.*, 25, 7 (1979), 623–630.

—— and C. A. Watts, "A Simulation Analysis of Tour-Shift Construction Procedures," *Management Sci.*, 28, 5 (1982), 520–532.

—— and M. J. Showalter, "Measuring the Impact of Part-Time Workers in Service Organizations," *J. Oper. Management*, 9, 2 (1990), 209–229.

Marsten, R. E., "The Design of the XMP Linear Programming Library," *Transactions on Mathematical Software*, 7, 4 (1981), 481–497.

Morris, J. G. and M. J. Showalter, "Simple Approaches to Shift, Days-Off and Tour Scheduling Problems," *Management Sci.*, 29, 8 (1983), 942–950.

Ritzman, L. P., L. J. Krajewski and M. J. Showalter, "The Disaggregation of Aggregate Manpower Plans," *Management Sci.*, 22, 2 (1976), 1372–1380.

Segal, M., "The Operator Scheduling Problem: A Network Flow Approach, *Oper. Res.*, 22, 4 (1974), 808–824.

Accepted by L. Joseph Thomas, former Departmental Editor; received June 1, 1992. This paper has been with the authors 1 month for 2 revisions.

MANAGEMENT SCIENCE
Vol. 54, No. 2, February 2008, pp. 295–309
ISSN 0025-1909 | EISSN 1526-5501 | 08 | 5402 | 0295

informs

DOI 10.1287/mnsc.1070.0774

Optimizing Call Center Staffing Using Simulation and Analytic Center Cutting-Plane Methods

Júlíus Atlason
THOR Development and Research, Reykjavík, Iceland, julius@throun.is

Marina A. Epelman
Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, Michigan 48109, mepelman@umich.edu

Shane G. Henderson
School of Operations Research and Information Engineering, Cornell University, Ithaca, New York 14853, sgh9@cornell.edu

We consider the problem of minimizing staffing costs in an inbound call center, while maintaining an acceptable level of service in multiple time periods. The problem is complicated by the fact that staffing level in one time period can affect the service levels in subsequent periods. Moreover, staff schedules typically take the form of shifts covering several periods. Interactions between staffing levels in different time periods, as well as the impact of shift requirements on the staffing levels and cost, should be considered in the planning. Traditional staffing methods based on stationary queueing formulas do not take this into account. We present a simulation-based analytic center cutting-plane method to solve a sample average approximation of the problem. We establish convergence of the method when the service-level functions are discrete pseudoconcave. An extensive numerical study of a moderately large call center shows that the method is robust and, in most of the test cases, outperforms traditional staffing heuristics that are based on analytical queueing methods.

Key words: simulation optimization; call centers; analytic center cutting-plane method
History: Accepted by Ger Koole, special issue editor; received October 29, 2004. This paper was with the authors $4\frac{1}{2}$ months for 1 revision. Published online in *Articles in Advance* December 11, 2007.

1. Introduction

A call center (also referred to as a contact center) is an important component of the operations of many organizations. A significant portion of the cost of operating a call center is staffing cost (Gans et al. 2003). In this paper, we describe a method for finding staff- (agent-) level requirements, while simultaneously selecting staff shifts that cover these requirements, that minimize cost while ensuring satisfactory customer service. An analytic center cutting-plane method guides the selection of staffing levels, and simulation is used to report service-level performance for each set of staffing levels.

This problem has received a great deal of attention in the literature, and therefore one can reasonably ask why there is a need for a computational tool of this sort. To answer that question, we first need to describe the overall staffing process. There are variations on the following theme (e.g., Castillo et al. 2007), but the essential structure is sequential in nature and is as follows (Mason et al. 1998).

1. *Forecasting*: Obtain forecasts of customer load over the planning horizon, which is typically one or two weeks long. The horizon is usually broken into short periods that are typically between 15 minutes and 1 hour long.

2. *Work requirements*: Determine the minimum number of agents needed during each period to ensure satisfactory customer service. Service is typically measured in terms of customer waiting times and/or abandonment rates in the queue.

3. *Shift construction*: Select staff shifts that cover the requirements. This problem is usually solved through a set-covering integer program; see Mason et al. (1998) for more details.

4. *Rostering*: Allocate employees to the shifts.

The focus in this paper is on Steps 2 and 3. Steps 1 and 4 are not considered further.

Step 2 is usually accomplished through the use of analytical results for simple queueing models. Green et al. (2001) coined the term SIPP (stationary, independent, period by period) to describe the general approach. In the SIPP approach, each period of the day is considered independently of other periods, the arrival process is considered to be stationary in that period, and one approximates performance in the period by a steady-state performance measure that is usually easily obtained from analytical results for particular queueing models. Heuristics are used to select input parameters for each period that yield a close match between the predictions and reality. For a wide

variety of queueing models, this procedure results in some form of the "square-root rule," which is a rule of thumb that provides surprisingly successful staffing level suggestions. See, e.g., Borst et al. (2004), Kolesar and Green (1998), and Jennings et al. (1996) for more on the square-root rule.

The SIPP approach is appealing from the standpoint of computational tractability and because of the insights it provides. However, there are cases where the SIPP approach does not do as well as one might hope (Green et al. 2001, 2003). For example, this can occur when the service times are relatively long; see Whitt (1991) for more on this point. Moreover, call centers can be somewhat complex in structure, and this complexity can make it difficult to identify a queueing model of the center that is both mathematically tractable and a reasonable match to the true system. In such cases, simulation is a viable alternative. Indeed, simulation is now increasingly used in Step 2; see §VIII of Mandelbaum (2003) for many examples.

Further motivation for the use of simulation involves the linkage between staffing decisions in adjacent periods. Boosting staffing levels in one period can often help in reducing workload in subsequent periods, so there can be linkage in performance between different periods. Such linkage can imply that there are multiple solutions to the work requirements problem that can offer valuable flexibility in Step 3. Traditional queueing approaches are not satisfactory in such cases, and then one turns to simulation or other numerical methods. Indeed, Green et al. (2001, 2003) solve a system of ordinary differential equations through numerical integration to get the "exact" results for their models to compare the performance of various heuristics.

Assuming that one uses simulation or some other numerical method to predict performance in the call center, one then needs to devise a method to guide the selection of potential staffing levels to be evaluated. There have been several suggestions in the literature, all of which explicitly capture the linkage between periods in an attempt to realize cost savings. Like Green et al. (2001, 2003), Ingolfsson et al. (2002) use numerical integration to compute service-level performance for a proposed set of staffing levels, and a genetic algorithm to guide the search. Ingolfsson et al. (2007) again use a numerical method to compute service-level performance, and integer programming to guide the search. Castillo et al. (2007) devise a method for randomly generating sets of staff shifts, then use simulation to evaluate the service-level performance of each set of generated staff shifts, and finally, plot the cost versus service level of the potential solutions to identify an efficient frontier. Atlason et al. (2004) use simulation to evaluate service level performance of a proposed set of shifts, and use integer programming in conjunction with Kelley's cutting plane method (Kelley 1960) to guide the search.

In Atlason et al. (2004), performance in each time period is measured as the fraction of calls responded to on time, and analysis relies on the assumption that service in a period is concave and componentwise increasing as a function of the staffing-level vector. To understand this assumption, consider a single-period problem. Increasing the staffing-level should lead to improved performance, i.e., an increasing fraction of calls responded to on time. Furthermore, one might expect "diminishing returns" as the staffing level increases, so that performance would be concave in staffing level. Empirical results suggest that this intuition is correct, at least for sufficiently high staffing levels. But for low staffing levels, the empirical results suggest that performance is increasing and convex in the staffing level. Therefore, performance appears to follow an "S-shaped" curve (Ingolfsson et al. 2007, Atlason et al. 2004) in one dimension.

This nonconcavity can cause the cutting-plane method of Atlason et al. (2004) to cut off feasible solutions, and the problem can be so severe as to lead to the algorithm suggesting impractical staffing plans. Nevertheless, the ability of the Atlason et al. (2004) approach to efficiently sift through the combinatorially huge number of potential staffing plans is appealing. One might ask whether there is a similar optimization approach that can satisfactorily deal with S-shaped curves and their multidimensional extensions. That is the subject of this paper. We replace the assumption of concavity with a weaker one (namely, pseudoconcavity), and use a different cutting-plane method that, together with additional techniques, successfully handles multidimensional extensions of the S-shaped curves alluded to above and seen in examples like that depicted in Figure 2; see §4.1 for more details.

Assuming that the service-level functions are pseudoconcave, an analytic center cutting-plane algorithm can be used to efficiently search the combinatorially large space of potential staffing plans. In essence, the algorithm works as follows. One begins with a polyhedron that contains an optimal solution to the staffing problem. At each stage, the algorithm selects a staffing plan that lies close to the analytic center of the polyhedron and runs a simulation at that point to determine service-level performance. Depending on the results of the simulation, an "optimality cut" or one or more feasibility cuts are added, thereby shrinking the polyhedron. The algorithm terminates when the polyhedron contains no integer points, or when the difference between upper and lower bounds on the optimal objective is sufficiently small.

We view the contributions of this paper as follows.

1. Under realistic assumptions on the service-level functions, we give an algorithm for solving to

optimality the combined Step 2–3 problem. The combined procedure explicitly addresses linkage between periods and the impact of shift-based scheduling.

2. We give conditions under which the algorithm provably converges.

3. We compare the proposed algorithm to what can be reasonably viewed as the current best practice to better understand the properties of the algorithm. This comparison must take place with a model of limited complexity so that the existing methods can be applied, but is nevertheless quite illuminating.

The numerical results in §5 show that the analytic center cutting-plane method outlined here outperforms, or at least equals, the SIPP heuristics in every case in which shift structure of staff scheduling is explicitly considered, which is the setting we are interested in. In that sense, it is a robust procedure that can be applied in a near black-box fashion. Of course, these extremely appealing properties have to be traded off against the computational cost of the procedure, which is not insignificant. We are actively considering methods for reducing the computational effort of the procedure; see §6.

The remainder of this paper is organized as follows. We formulate the call center staffing problem in §2. In §3, we review cutting-plane methods for continuous problems with pseudoconcave constraints. Section 4 describes modifications for discrete problems, such as the call center staffing problem, and proves that the algorithm converges. We compare the proposed algorithm with the SIPP methods described in Green et al. (2001, 2003) in §5. Section 6 concludes the paper and discusses future research questions.

2. Problem Formulation

In this section, we formulate the call center staffing problem of minimizing staffing cost over a planning horizon while maintaining an acceptable level of service in each of a number of time periods, with particular emphasis on the use of simulation to estimate the service levels. We say that service is acceptable in a period if the long-run percentage of calls that arrive in this period and are answered within a certain time limit τ meets or exceeds a threshold π. This is the usual "π percent of calls are answered within τ seconds" requirement that is something of an industry standard, cf. the "80/20 rule" for which π is 80% and τ is 20 seconds.

Define the staffing-level vector as $y = (y_1, \ldots, y_p)^T$, where p is the number of time periods in the planning horizon, y_i is the number of agents working in period i, $i = 1, \ldots, p$, and $(\cdot)^T$ denotes the transpose operation. Let N_i be the random number of calls that are received in period i, and let $S_i(y)$ be the random number of those calls that are satisfactorily handled, $i = 1, \ldots, p$. We can express the service-level constraint $E[S_i(y)]/E[N_i] \geq \pi$ as

$$g_i(y) = ES_i(y) - \pi EN_i \geq 0, \quad i = 1, \ldots, p.$$

Here, $g_i(y)$ gives the expected number of successful services over and above the required expected number in period i. (Note that requiring that $E[S_i(y)/N_i] \geq \pi$ is not a valid way to enforce the constraint because it gives more weight to service on days where the call volume is low. Also note that we include a separate service-level constraint for each period to ensure uniform quality of service throughout the planning horizon, which is consistent with industry practices.)

For a staffing-level vector y, there is an associated cost of covering the staffing levels with tours. The term "tour" refers to a collection of periods that a single employee works over the planning horizon. A tour must agree with all provisions of employee contracts, such as rules on meal breaks and a limit on the number of hours an agent can work over the planning horizon. All feasible tours are enumerated prior to problem formulation and stored in the tour matrix A such that $A_{ij} = 1$ if tour j includes period i, and zero otherwise. Also, the jth component of the cost vector c is the cost of the jth tour. We let x be a vector with jth component x_j giving the number of employees working the jth tour. We assume that there are m feasible tours, therefore $x \in \mathbb{Z}_+^m$. Using this notation, the minimal cost of covering work requirement vector y is

$$\begin{aligned} f(y) = \min \ & c^T x \\ \text{s.t.} \ & Ax \geq y, \\ & x \geq 0 \text{ and integer.} \end{aligned} \tag{1}$$

We assume that every period is covered by at least one tour, i.e., for every i there is at least one j such that $A_{ij} = 1$. It then follows that (1) is feasible for any y.

The call center staffing problem can now be formulated as

$$\begin{aligned} \min \ & f(y) \\ \text{s.t.} \ & g_i(y) \geq 0 \quad \text{for } i = 1, \ldots, p, \\ & y \geq 0 \text{ and integer.} \end{aligned} \tag{2}$$

Recall that we cannot compute the (vector-valued) service-level function $g(y)$ exactly, and instead use simulation. The call center staffing problem is then a simulation optimization problem. As in Atlason et al. (2004), we adopt "sample-average approximation" (see Shapiro 2003, Kleywegt et al. 2001) for solving such problems. This approach, specialized to our setting, is as follows. One first generates the simulation input data sets for n independent replications

of the operations of the call center over the planning horizon. Each data set includes call arrival times and service times, and other input data needed to simulate operations of the call center at hand, if appropriate, e.g., call abandonment times and so forth. The simulation data sets are fixed, leading to a deterministic optimization problem that chooses staffing levels so as to minimize staffing cost, while ensuring that average service *computed only over the generated realizations* is satisfactory. We can then attempt to identify or develop deterministic optimization methods to solve the resulting problem.

Suppose that service-level functions $g_i(y)$, $i = 1, \ldots, p$, are estimated by corresponding sample averages $\bar{g}_i(y; n)$, where n is the sample size used. The sample average approximation of the call center staffing problem is then

$$\begin{aligned} \min\ & f(y) \\ \text{s.t. } & \bar{g}_i(y; n) \geq 0 \quad \text{for } i = 1, \ldots, p, \\ & y \geq 0 \text{ and integer.} \end{aligned} \tag{3}$$

Proposition 1 below gives conditions under which solutions to (3) converge to those of the true problem (2) as $n \to \infty$. We first make the following natural assumption about the cost vector.

Assumption 1. *The cost vector c is positive and integer valued.*[1]

Assumption 1 implies that $f(y)$ is integer valued. Moreover, because $c > 0$, the z-level set of f,

$$\{y \geq 0 \text{ and integer: } \exists x \geq 0 \text{ and integer}, Ax \geq y, c^T x \leq z\},$$

is finite for any $z \in \mathbb{R}$.

Let Y^* denote the set of optimal solutions to the true problem (2).

Proposition 1. *Suppose that Assumption 1 holds. Suppose further that for each y, $\bar{g}_i(y; n) \to g_i(y)$ as $n \to \infty$ with probability one. Finally, suppose that there is an optimal solution $y^* \in Y^*$ that is "strictly feasible," i.e., $g_i(y^*) > 0$ for all $i = 1, \ldots, p$. If y_n^* is an optimal solution to (3) for each n, then $y_n^* \in Y^*$ for n sufficiently large, with probability one.*

This result is easily proved using techniques very similar to those used in Atlason et al. (2004), and therefore we omit the proof. The requirement of a "strictly feasible" optimal solution is easily justified in practice: it is almost surely impossible for the service-level function g_i to be exactly zero when y takes on only discrete values.

Proposition 1 establishes the validity of the sample average approximation approach in our setting. Much more can be said about the convergence of solutions of the sample average approximation problem and their objective values, but that is not the emphasis of this paper. We refer the interested reader to Atlason et al. (2004) and Kleywegt et al. (2001).

3. The Analytic Center Cutting-Plane Method

In this section, we state a version of the traditional analytic center cutting-plane method (ACCPM) to fix ideas and provide a departure point for a reader unfamiliar with cutting-plane methods in continuous optimization.

There are many cutting-plane methods for solving convex optimization problems, including what may be termed "boundary methods," such as Kelley's algorithm (Kelley 1960) and its extensions (e.g., Westerlund and Pörn 2002), and interior point methods, recently reviewed by Mitchell (2003). We will focus our attention on one of the latter, namely, the ACCPM, which was first implemented in du Merle (1995), as pointed out in Elhedhli and Goffin (2003). The ACCPM has proven effective in terms of both theoretical complexity (Atkinson and Vaidya 1995, Nesterov 1995, Goffin et al. 1996, Mitchell 2003) and practical performance on a variety of problems (Bahn et al. 1995, Mitchell 2000, and other references in Mitchell 2003). Software packages implementing the method are available (e.g., Peton and Vial 2001).

Many versions of the ACCPM for convex feasibility and optimization problems have been explored in the literature. The description we chose below borrows from Nesterov (1995), du Merle et al. (1998), and Mitchell (2003), among others.

Consider an optimization problem in the following general form:

$$\begin{aligned} \min\ & b^T y \\ \text{s.t. } & y \in Y, \end{aligned} \tag{4}$$

where $Y \subset \mathbb{R}^n$ is a convex set, and $b \in \mathbb{R}^n$. (A problem of minimizing a general convex function over a convex set can be easily represented in this form.) To simplify the presentation, assume that the set Y is bounded and has a nonempty interior.

To apply the ACCPM (or any other cutting-plane algorithm), the feasible region Y needs to be described by a separation oracle. Such an oracle will, given an

[1] The integrality part of Assumption 1 can be satisfied in practice by multiplying by the appropriate power of 10 if necessary, e.g., by 100 if the cost is in dollars and cents. This does not change the computational complexity of the problem because the number of significant digits to represent the data remains the same. In this case, attention should be paid to the choice of the weight parameter w defined in §3 to get a significant reduction in the objective value between the associated optimality cuts.

input $\hat{y} \in \mathbb{R}^n$, either correctly assert that $\hat{y} \in Y$, or otherwise return a nonzero vector $q \in \mathbb{R}^n$, such that

$$q^T(y - \hat{y}) \geq 0 \quad \forall y \in Y,$$

i.e., produce a *feasibility cut*. (Depending on how the set Y is described, the oracle might produce a deep or shallow cut, which has the same form as the constraint above, but a positive or negative right-hand side, respectively.)

We now describe a typical iteration of the ACCPM. At the beginning of an iteration, we have available a finite upper bound z on the optimal value of (4), and a polyhedron $P = \{y \in \mathbb{R}^n: Dy \geq d\}$ that is known to contain the set Y. Here, $D \in \mathbb{R}^{r \times n}$ and $d \in \mathbb{R}^r$ for some finite r. We first compute the (weighted) analytic center of the set $P \cap \{y \in \mathbb{R}^n: b^T y < z\}$ (for ease of presentation, we assume that the set $P \cap \{y \in \mathbb{R}^n: b^T y < z\}$ is bounded), defined as the solution of the convex optimization problem

$$\min_{y \in \text{int}(P \cap \{y \in \mathbb{R}^n: b^T y \leq z\})} \left\{ -w \log(z - b^T y) - \sum_{l=1}^{r} \log(D_l y - d_l) \right\}, \tag{5}$$

where D_l is the lth row of D and $w > 0$ is a weight constant that affects the convergence rate of the algorithm (see, for example, du Merle et al. 1998). The set $P \cap \{y \in \mathbb{R}^n: b^T y < z\}$ is often referred to as the *localization set* because it contains all feasible solutions with objective function values lower than z.

Finding a solution to (5) with a high degree of precision is a relatively simple task from a practical standpoint and can be done, e.g., via Newton's method. Let $\hat{y}$ be the analytic center found. Next, the oracle is called with $\hat{y}$ as the input. If $\hat{y} \in Y$, then, by construction, $b^T \hat{y} < z$, and the upper bound is lowered by taking $z := b^T \hat{y}$. Otherwise, if $\hat{y} \notin Y$, the oracle will produce a vector q providing a feasibility cut, which is then added to the description of the polyhedron P. The procedure is then repeated. A slightly more detailed description of the algorithm is presented in Figure 1.

Intuitively, the algorithm's efficiency stems from the fact that at each iteration the cut being added passes through the analytic center of the localization set, which is often located near a geometric center. Thus, the volume of the localization set reduces rapidly with each iteration.

Suppose that the feasible set Y is specified by $Y = \{y \in \mathbb{R}^n: g_i(y) \geq 0, i = 1, \ldots, p\}$, where the functions $g_i(y)$, $i = 1, \ldots, p$, are pseudoconcave, as defined below.

Definition 1 (Cf. Definition 3.5.10 in Bazaraa et al. 1993). Let $h: S \to \mathbb{R}$ be differentiable on S, where S is a nonempty open set in $\mathbb{R}^n$. The function h is said to be *pseudoconcave* if for any $\hat{y}, y \in S$, with $\nabla h(\hat{y})^T(y - \hat{y}) \leq 0$, we have $h(y) \leq h(\hat{y})$. Equivalently, if $h(y) > h(\hat{y})$, then $\nabla h(\hat{y})^T(y - \hat{y}) > 0$.

Figure 1 The Analytic Center Cutting-Plane Method (ACCPM) for Problem (4)

Initialization Start with a polyhedron $P^0 := \{y \in \mathbb{R}^n: D^0 y \geq d^0\}$ such that $Y \subset P^0$, a solution $y^* \in Y$, and an upper bound $z^0 := b^T y^*$. Let $w^0 > 0$, and set the iteration counter $k := 0$.

Step 1. If termination criterion is satisfied, then stop, and return y^* as a solution. Otherwise, solve Problem (5) with $w = w^k$, $z = z^k$, and $P = P^k := \{y \in \mathbb{R}^n: D^k \geq d^k\}$; let y^k be the solution.

Step 2a. If $y^k \in Y$, then let $z^{k+1} := b^T y^k$, $y^* := y^k$, $D^{k+1} := D^k$, and $d^{k+1} := d^k$.

Step 2b. If $y^k \notin Y$, then generate one or more feasibility cuts at y^k. Update D^k and d^k to include the new constraints, and let D^{k+1} and d^{k+1} represent the new constraint set. Let $z^{k+1} := z^k$.

Step 3. Compute w^{k+1} and let $k := k + 1$. Go to Step 1.

Pseudoconcavity of a function is a weaker property than concavity; see also Bazaraa et al. (1993, Figure 3.12).

With Y in the above form, the feasibility cut at point $\hat{y} \notin Y$ which violates the ith constraint can be specified as

$$\nabla g_i(\hat{y})^T(y - \hat{y}) \geq 0 \tag{6}$$

because any solution y that satisfies $g_i(y) \geq 0$ also satisfies $g_i(y) > g_i(\hat{y})$.

4. A Cutting-Plane Method for Discrete Problems

In this section, we describe how the ACCPM algorithm of §3 can be modified to solve the sample average approximation (3) of the call center staffing problem (2).

The most significant modification to the ACCPM for continuous problems is needed to take into account the fact that the feasible region of (3) is no longer a convex, or even connected, because of set the integrality restriction on the variables. Cutting-plane algorithms for nonlinear mixed-integer programs have been explored in the past; see, for example, Westerlund and Pörn (2002). However, in this and other similar papers, it is assumed that the constraint functions are in fact differentiable functions of continuous variables; the integrality restrictions on the variables are, in a sense, exogenous. In such a setting, the concept of a convex (continuous) nonlinear relaxation of the integer program is straightforward, and feasibility cuts are generated simply using the gradients of these continuous functions. In our setting, however, the service-level functions and their sample average approximations are not defined at noninteger values of y, and devising their continuous extension, especially one that is easy to work with from the algorithmic point of view, is nontrivial at best. Therefore, we take a different approach in adapting the ACCPM to the discrete case.

As far as we know, the use of ACCPM as a solution method for nonlinear integer programs has not been reported, although the method has been successfully used to solve the linear relaxation subproblems in branching algorithms for integer programs (see, for example, Mitchell 2000 and Elhedhli and Goffin 2003, among many others).

In §4.1, we extend the notion of pseudoconcavity to functions of integer variables, and show how feasibility cuts can be generated assuming that the functions $\bar{g}_i(y; n)$, $i = 1, \ldots, p$, are, in fact, discrete pseudoconcave. This leads to an ACCPM method for (mixed) integer programming problems satisfying the pseudoconcavity assumption; the algorithm is applicable to the types of problems considered in Westerlund and Pörn (2002), for example. We also discuss whether the S-shaped form of the service-level functions in the call center staffing problem is consistent with this assumption, and propose alternative feasibility cuts at points where it is violated.

Section 4.2 discusses other modifications of the original algorithm for Problem (3) and details of our implementation. Section 4.3 gives a proof of convergence.

4.1. Discrete Pseudoconcave Functions and Feasibility Cuts

We begin by defining the notions of a discrete convex set and a discrete pseudoconcave function. We denote the convex hull of the set C by $\text{conv}(C)$.

Definition 2. We say that the set $C \subseteq \mathbb{Z}^n$ is a *discrete convex set* if $C = \text{conv}(C) \cap \mathbb{Z}^n$, i.e., the set C equals the set of integer points in $\text{conv}(C)$.

Definition 3. Let $C \subseteq \mathbb{Z}^n$ be a discrete convex set and $h\colon C \to \mathbb{R}$. Then, h is *discrete pseudoconcave* on C if for any $\hat{y} \in C$, there exists a vector $q(\hat{y}) \in \mathbb{R}^n$ such that for any $y \in C$,

$$q(\hat{y})^T(y - \hat{y}) \le 0 \;\Rightarrow\; h(y) \le h(\hat{y}). \tag{7}$$

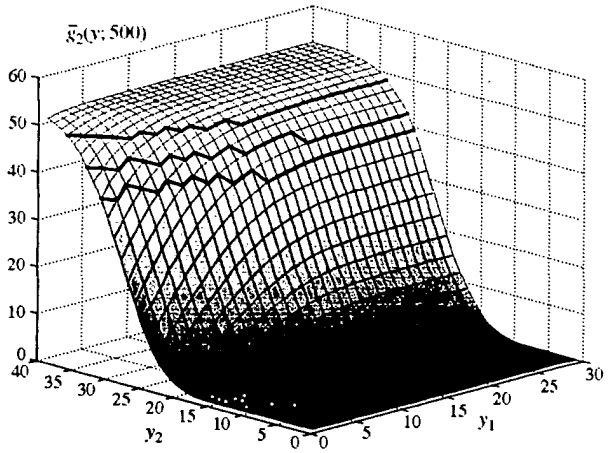
Equivalently, if $h(y) > h(\hat{y})$, then $q(\hat{y})^T(y - \hat{y}) > 0$. We call such a vector $q(\hat{y})$ a *pseudogradient* of h at $\hat{y}$.

Discrete pseudoconcavity can be viewed as a relaxation of the concave extensible function property defined in Murota (2003, p. 93). Note also the similarity between the above definition and Definition 1. For a differentiable pseudoconcave function, the gradient at point $\hat{y}$ plays the role of a pseudogradient $g(\hat{y})$ in condition (7).

If the functions $\bar{g}_i(y; n)$ in (3) are discrete pseudoconcave, then a feasibility cut at an integer point $\hat{y}$ that violates the ith constraint can be obtained in the form $\bar{q}^i(\hat{y}; n)^T(y - \hat{y}) \ge \epsilon$, where $\bar{q}^i(\hat{y}; n)$ is the pseudogradient of $\bar{g}_i(\cdot; n)$ at $\hat{y}$, and $\epsilon > 0$ is sufficiently small.

Are the service-level functions we consider indeed discrete pseudoconcave? As an illustrative example, consider a simple call center with two 30-minute periods. The service times are exponential with a mean of 15 minutes. The arrival process is a nonhomogeneous Poisson process with rate varying between 42 and 120 calls per hour with an average load of 18 in period 1 and 25 in period 2 (*load* is the arrival rate divided by the service rate). Figure 2 shows the number of calls answered on time (here, $\tau = 90$ seconds) in period 2 as a function of the staffing levels in periods 1 and 2. (Note that this is equivalent to $\bar{g}$ with $\pi = 0$.) The number of servers ranges from 1 to 30 in period 1 and from 1 to 40 in period 2. The function appears to follow a multidimensional extension of an S-shaped curve discussed in §1 (see also Ingolfsson et al. 2007, Atlason et al. 2004).

Figure 2 **The Sample Average (Sample Size $n = 500$) of the Number of Calls Answered on Time in Period 2 as a Function of the Staffing Levels in Periods 1 and 2**



Notes. The white dots on the bottom and the black dot on the top represent points where a pseudogradient does not exist. The thicker black lines show the boundaries of the satisfactory service levels for π equal to 70%, 80%, and 90%.

To verify that the function is discrete pseudoconcave, one can solve for each point, the linear (feasibility) program

$$\begin{aligned} \min\ & 0 \\ \text{s.t. } & q^T(y - \hat{y}) \ge 1 \quad \forall y \in \{y\colon \bar{g}_i(y; n) > \bar{g}_i(\hat{y}; n)\}, \\ & q \ge 0. \end{aligned} \tag{8}$$

Without the nonnegativity constraint, Problem (8) has a solution q if and only if a pseudogradient exists, and then q is such a pseudogradient. We added this constraint because the service-level function can reasonably be assumed to be nondecreasing in each component of y. The number of constraints in Problem (8) can be very large and it cannot be constructed without simulating the service-level function for all practical values of y. In practice, however, one can solve the LP using only the points visited by the algorithm to

check whether they are consistent with the assumption of pseudoconcavity.

The results of Problem (8) for the function in Figure 2 indicate that the function is very close to being discrete pseudoconcave. There are only a few points in the entire domain where a pseudogradient does not exist. These points are depicted in Figure 2 by the white dots on the bottom and a single black dot (at $(y_1, y_2) = (10, 37)$) on top of the graph. Note that these points are in relatively flat areas of the graph. In the next section, we describe a simple way to advance the algorithm at such points, which we successfully used in the implementation of our cutting-plane method.

Moreover, in a flat area, for a finite sample size, each additional call answered on time can have a significant effect on the order of the values of the service-level function. Therefore, it seems reasonable to assume that this situation would be alleviated for larger sample sizes, and in particular, to assume that the *expected* service-level function is discrete pseudoconcave. Thus, to prove the convergence results in §4.3, we assume that the sample averages of the service-level functions are discrete pseudoconcave.

4.2. A Simulation-Based Cutting-Plane Method for the Call Center Staffing Problem with Pseudoconcave Service-Level Functions

In this subsection, we describe our modification of the ACCPM for Problem (3). At the beginning of a typical iteration, we have available a feasible solution y^* of (3), and an upper bound $z = f(y^*)$ on the optimal value of the problem. The point y^* is the best feasible solution found by the algorithm so far. We also have available a polyhedron $P = \{y \in \mathbb{R}^p\colon\ y \geq 0, Dy \geq d\}$ that is known to contain the feasible region.

Suppose that y^*, z, and P are as above. If y^* is not an optimal solution, then, because $f(y)$ takes on integer values, the localization set

$$\{y \in P\colon f(y) \leq z - 1, y \text{ integer}\}$$

is nonempty and contains all feasible solutions with objective values better than z. The localization set is empty precisely if y^* is an optimal solution. This observation provides grounds for the termination criteria we specify in our algorithm.

Computing the next iterate. First, we find the analytic center of a polyhedral relaxation of the localization set. In particular, we solve the following optimization problem:

$$\begin{aligned} \min\ & -w\log(z - a - c^T x) - \sum_{i=1}^{p}\log y_i \\ & - \sum_{j=1}^{r}\log(D_j y - d_j) \\ \text{s.t. } & Ax \geq y, \\ & x \geq 0, \end{aligned} \tag{9}$$

where the constant $a \in (0, 1)$ ensures that only points with an objective value better than z are considered. Due to Assumption 1, the effective feasible region of (9) is bounded; hence the problem has an optimal solution as long as it has a feasible point in the effective domain of the objective. If (9) has no solution, the algorithm terminates with y^* as an optimal solution; otherwise, let (y^{ac}, x^{ac}) be a solution of (9). Here, $w > 0$ is the weight constant; we discuss later how this constant is determined in the algorithm.

Note that the analytic center found in the previous step is almost certainly not integer, and rounding y^{ac} to the nearest integer can result in a point outside of the localization set. To capitalize on the centrality of the analytic center in the localization set, we instead find the closest integer point in the effective feasible region of (9), i.e., solve

$$\begin{aligned} \min\ & \|y - y^{ac}\| \\ \text{s.t. } & Ax \geq y \\ & c^T x \leq z - 1 \\ & Dy \geq d \\ & x, y \geq 0 \text{ and integer.} \end{aligned} \tag{10}$$

If (10) is infeasible, the algorithm terminates with y^* as an optimal solution; otherwise, let $(\hat{y}, \hat{x})$ be the solution of (10) and choose $\hat{y}$ as the next iterate. Here, $\|y - y^{ac}\|$ is a measure of the distance between y and y^{ac}. If we choose the L_1-norm as the measure, i.e., $\|y - y^{ac}\| = \sum_{i=1}^{p} |y_i - y_i^{ac}|$, then (10) is a linear integer program. We discuss the computational requirements of solving this problem at each iteration when we talk about the overall computational effort in relation to the computational experiments.

Estimating the service levels. Next, we compute $\bar{g}_i(\hat{y}; n)$ for all i via simulation.

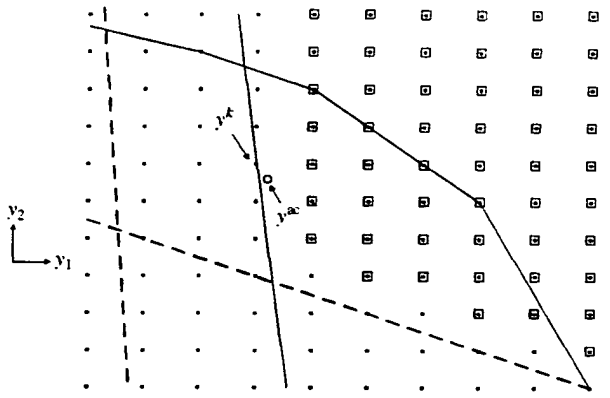
Adding an optimality cut. If $\bar{g}_i(\hat{y}; n) \geq 0$ for all i, then $\hat{y}$ satisfies the service-level requirements. Because $c^T\hat{x} \leq z - 1$, $\hat{y}$ is contained in the localization set, i.e., it is the best staffing level so far. Note that $c^T\hat{x}$ is not necessarily the cost associated with staffing level $\hat{y}$ because $c^T x$ is not being minimized in (10). To compute the cost associated with $\hat{y}$, we instead solve (1) to get $f(\hat{y})$ and update $z := f(\hat{y})$.

Adding a feasibility cut. If $\bar{g}_i(\hat{y}; n) < 0$ for some i, then we add a feasibility cut for each i such that $\bar{g}_i(\hat{y}; n) < 0$. In particular, we estimate a pseudogradient, $\bar{q}^i(\hat{y}; n)$, of $\bar{g}_i(\cdot; n)$ at $\hat{y}$ (we will discuss techniques for estimating the pseudogradients in §5.2). If $\bar{q}^i(\hat{y}; n) \neq 0$, we add a feasibility cut of the form

$$\bar{q}^i(\hat{y}; n)^T y \geq \bar{q}^i(\hat{y}; n)^T\hat{y} + \epsilon \tag{11}$$

for some small constant $\epsilon > 0$ (we discuss the role of ϵ in the discussion of the convergence of the method in

Figure 3 **Illustration of the Feasible Region and an Iterate in the SACCPM**



§4.3). If $\bar{q}^i(\hat{y}; n) = 0$, the feasibility cut takes the form of a lower bound on the number of servers (see the discussion in §5.2). We update D and d to reflect the cuts added.

The above procedure is then repeated. An illustration of the localization set and the feasible regions and solutions of Problems (9) and (10) in each iteration is shown in Figure 3. The picture shows points in the space of the y variables in the case $p = 2$. The shaded area represents the localization set $\{y \in P: f(y) \le z - 1\}$ (ignoring integer constraints). The squares represent the points that are feasible for the sample average approximation problem (3). The thick solid line segments result from an optimality cut $f(y) \le z - 1$ and the dotted lines represent the feasibility constraints $Dy \ge d$. Point y^{ac} is the solution of the analytic center problem, and y^k is the closest integer in the localization set. The line just to the right of y^k represents a feasibility cut that would be added in this iteration.

We summarize the simulation-based analytic center cutting-plane method (SACCPM) for the call center staffing problem in Figure 4. (To shorten the presentation, the description of the algorithm in Figure 4 is only specified for the case when the constraint functions $\bar{g}_i(\cdot; n)$ are in fact discrete pseudoconcave.)

The weight parameter w on the optimality cut can be increased to "push" the weighted analytic center away from the optimality cuts. There are no theoretical results on how to choose the weights, but some computational experiments have been done to test different values of the weights, and in fact, weight parameters on the feasibility cuts (Goffin et al. 1992, du Merle et al. 1998). The choice of the weights is problem and data dependent (see §5.2 for the particular choice used in our implementation).

In practice, it is useful to maintain a lower bound on the optimal value of Problem (3) throughout the algorithm, in addition to the upper bound z. In particular, the algorithm can be terminated as soon as

Figure 4 **The Simulation-Based Analytic Center Cutting-Plane Method (SACCPM)**

Initialization Start with a solution y^0 feasible to Problem (3). Let $z^0 := f(y^0)$, and $y^* := y^0$. Let $P^0 = \{y \ge 0: D^0 y \ge d^0\}$, where D^0 and d^0 are empty. Choose an $\epsilon > 0$, $a \in (0, 1)$ and $w^0 > 0$. Let $k := 0$.

Step 1. Solve Problem (9) with $w = w^k$, $z = z^k$, and $P = P^k$. If (9) does not have a (feasible) solution, then terminate with y^* as the optimal solution and z^k as the optimal value; otherwise let y^{ac} be the solution of (9).

Step 2. Solve Problem (10) with $z = z^k$ and $P = P^k$. If (10) is infeasible, then terminate with y^* as the optimal solution and z^k as the optimal value; otherwise let y^k be the optimal solution of (10) found.

Step 3a. If $\bar{g}_i(y^k; n) \ge 0\ \forall i = 1, \ldots, p$, let $z^{k+1} := f(y^k)$, $y^* := y^k$, $D^{k+1} := D^k$, and $d^{k+1} := d^k$.

Step 3b. If $\bar{g}_i(y^k; n) < 0$ for some $i = 1, \ldots, p$, then add the constraint $\bar{q}^i(y^k; n)^T y \ge \bar{q}^i(y^k; n)^T y^k + \epsilon$ for each i such that $\bar{g}_i(y^k; n) < 0$. Update D^k and d^k to include the added inequalities, and let $P^{k+1} = \{y \ge 0: D^{k+1} y \ge d^{k+1}\}$ represent the new constraint set. Let $z^{k+1} := z^k$.

Step 4. Compute w^{k+1} and let $k := k + 1$. Go to Step 1.

the gap between the upper and lower bounds is sufficiently small, indicating that the current "incumbent" y^* is a sufficiently good solution of the problem. A lower bound can be found as the optimal objective value of

$$\begin{aligned} \min\ & f(y) \\ \text{s.t. } & Dy \ge d, \\ & y \ge 0 \text{ and integer}, \end{aligned} \tag{12}$$

or, for a weaker but easier to compute bound, of the linear programming relaxation of (12).

4.3. Convergence of the SACCPM

In this section, we give conditions under which the SACCPM terminates with y^* as an optimal solution of (3). First, we argue that the algorithm does indeed terminate and then we show that y^* is an optimal solution of (3) at termination. To prove the results, we make the following two assumptions.

ASSUMPTION 2. *The functions $\bar{g}_i(y; n)$ are discrete pseudoconcave in y for all $i = 1, \ldots, p$.*

ASSUMPTION 3. *In the implementation of the SACCPM, $\bar{q}^i(\hat{y}; n)$ is a pseudogradient of $\bar{g}_i(\cdot; n)$ at $\hat{y}$.*

We also define the sets

$$\begin{aligned} \Gamma &:= \{y \ge 0 \text{ and integer}: f(y) \le z^0\}, \\ \Psi &:= \Gamma \cap \{y: \bar{g}_i(y; n) \ge 0\ \forall i = 1, \ldots, p\}, \\ \Upsilon &:= \Gamma \setminus \Psi, \text{ and} \\ I(y) &:= \{i: \bar{g}_i(y; n) < 0\}. \end{aligned}$$

In words, Γ is the set of points that are potentially visited by the algorithm, and it contains the set of

optimal solutions. The set Ψ is the set of points in Γ that are feasible for the sample average approximation problem (3). The set Υ is the set of points in Γ that are not feasible for the sample average approximation problem (3). The set $I(y)$ is the set of periods in which the sample average of the service-level function is not acceptable given the staffing levels y. The following lemma says that all solutions in Ψ satisfy potential feasibility cuts (11) for some appropriately chosen ϵ.

LEMMA 1. *Let $\bar{q}^i(\hat{y}; n)$ be a pseudogradient of $\bar{g}_i(\cdot; n)$ at $\hat{y}$ and suppose that Assumptions* 1 *and* 2 *hold. Then, there exists an $\tilde{\epsilon} > 0$ such that $\bar{q}^i(\hat{y}; n)^T(y - \hat{y}) \geq \tilde{\epsilon}$ $\forall y \in \Psi$, $\hat{y} \in \Upsilon$, $i \in I(\hat{y})$.*

PROOF. Let $\hat{y} \in \Upsilon$ be fixed. Suppose that $\bar{q}^i(\hat{y}; n)^T \cdot (y - \hat{y}) \leq 0$ for some $y \in \Psi$ and $i \in I(\hat{y})$. Then, $\bar{g}_i(y; n) \leq \bar{g}_i(\hat{y}; n) < 0$, where the first inequality follows by Assumption 2 and the second inequality follows because $i \in I(\hat{y})$. This is a contradiction because $y \in \Psi$, and therefore $\bar{q}^i(\hat{y}; n)^T(y - \hat{y}) > 0$ $\forall y \in \Psi$, $i \in I(\hat{y})$.

Let $\epsilon(\hat{y}) = \min_{i \in I(\hat{y})} \min_{y \in \Psi} \bar{q}^i(\hat{y}; n)^T(y - \hat{y})$. The set Ψ is finite by Assumption 1; therefore, the inner minimum is attained for some $y \in \Psi$. Because $I(\hat{y})$ is also finite, the outer minimum is also attained for some $i \in I(\hat{y})$. Therefore, $\epsilon(\hat{y}) > 0$.

Finally, let $\tilde{\epsilon} = \min_{\hat{y} \in \Upsilon} \epsilon(\hat{y})$. Then, $\tilde{\epsilon} > 0$ because Υ is finite. □

THEOREM 1. *Suppose that* (3) *has an optimal solution and that Assumptions* 1, 2, *and* 3 *hold. Furthermore, let $0 < \epsilon \leq \tilde{\epsilon}$, where $\tilde{\epsilon}$ is as in Lemma* 1. *Then, the SACCPM terminates in a finite number of iterations returning y^*, which is an optimal solution of* (3).

PROOF. The SACCPM only visits points that are feasible solutions of (10). The set of feasible solutions of (10) at every iteration is a subset of Γ and therefore is finite by Assumption 1. Suppose that the kth iterate y^k is the same as a previous iterate y^j for some $j < k$. If y^j is in Ψ, then $f(y^k) \leq z - 1 \leq f(y^j) - 1$, where the second inequality follows because z is the cost of the best feasible solution visited before iteration k. This is a contradiction, therefore y^j is not in Ψ. On the other hand, if y^j is in Υ, then $\bar{g}_i(y^j; n) < 0$ for some $i \in I(y^j)$. Because y^k is in P, $\bar{q}^i(y^j; n)^T(y^k - y^j) \geq \epsilon > 0$, which is also a contradiction and y^k can therefore not be equal to y^j for any $j < k$. Therefore, the SACCPM does not visit any point in Γ more than once. Because Γ is a finite set, the algorithm is finite.

By Lemma 1, the feasibility cuts never cut off any of the feasible solutions of (3). At termination, Problem (9) and/or Problem (10) does not have a feasible solution, which means that all the feasible solutions for (3) have an objective value greater than $z - 1$. But y^* is feasible for (3) and has an objective value of z and is, therefore, optimal, because $f(y)$ is integer by Assumption 1. □

The theorem says that there exists an $\tilde{\epsilon}$ such that the algorithm terminates with an optimal solution of (3) if $0 < \epsilon \leq \tilde{\epsilon}$. In practice, the value of $\tilde{\epsilon}$ is unknown, but in general, a "small" value for ϵ should be chosen as in §5.4.

5. Numerical Results

In this section, we give an implementation of the SACCPM for a call center with time-varying arrival rates. In §5.1, we describe the example that we use for the numerical experiments. In §5.2, we discuss how to compute, or estimate, the pseudogradients, and we also describe how the algorithm was implemented. We compare the results with staffing levels obtained by analytical queueing methods (see §1 for references). Analytical queueing methods based on the Erlang C formula (14) are widely used to determine staffing levels in call centers (Cleveland and Mayben 1997). In Green et al. (2001, 2003), several heuristics for improving the performance of the basic Erlang C model are evaluated and we give a summary of these heuristics in §5.3. We believe that the methods in Green et al. (2001, 2003) are among the best analytical methods for determining required staffing levels to answer a minimum fraction of calls before a threshold time limit. Therefore, we use these methods as benchmarks for our method. The results of our experiments are in §5.4 and we comment on the computational requirements of the SACCPM in §5.5.

5.1. Example

Our test model is similar to the models used in Green et al. (2001, 2003), which are call centers that can be modeled as $M(t)/M/s(t)$ queueing systems. In the Green et al. (2001) paper, a call center operating 24 hours a day and seven days a week is studied, while the subject of the Green et al. (2003) paper is call centers with limited hours of operation. We consider a call center with limited hours of operation that is open from 6 A.M. to 12 A.M.

The call center has the following additional characteristics. The planning horizon consists of a single day's operation. The 18-hour planning horizon is broken into 72 time periods, each of length 15 minutes. In each period, 80% of calls should be answered immediately. This is equivalent to setting $\tau = 0$ and $\pi = 0.8$. The service times are independent exponential random variables with rate μ; see Table 1. The size of the system is measured by average load R, which is the average arrival rate divided by the service rate. The arrival process on any given day is a nonhomogeneous Poisson process with a continuous, piecewise-linear arrival rate defined at the end of period i by $\lambda_i = \lambda(1 + \theta \sin(2\pi(t_i - 6)/18))$, where t_i is

Table 1 The Parameter Settings in the Experiments

Parameter	Low	High
μ	4 calls/hour	16 calls/hour
R	8	32
θ	0.25	0.75
Shifts	Yes	No

the end time of period i and is measured in the hour of the day (e.g., $t_0 = 6$ and $t_{72} = 24$). The average daily arrival rate is $\lambda = R\mu$ and θ is the relative amplitude. Calls are answered in the order they are received. When there is a reduction in the number of servers at the end of a period, outgoing servers complete calls that they are already servicing before ending their shifts. A new call cannot enter service until there are fewer calls in service than there are servers for the new period.

We study the performance both in the presence of shift constraints and when there are no shift constraints. When there are no shift constraints, the tour matrix A is the identity matrix in $\mathbb{R}^p$, and the cost for each tour is equal to one agent-period, i.e., the total staffing cost equals the total number of servers over all 72 periods. We assume that the shift constraints are such that each tour covers six hours, or 24 periods. The tours can only start on the hour and no later than six hours before the end of the day. This results in 13 tours: 6 A.M.–12 P.M., 7 A.M.–1 P.M., ..., 5 P.M.–11 P.M., and 6 P.M.–12 A.M. There are no meal-type breaks incorporated in the shifts, but they can easily be included by changing the appropriate values in the tour matrix A. The cost for each tour is equal to 24 agent-periods per tour.

Note that we are yet to specify a value for μ, R, and θ. In the experiments, we study how the SACCPM performs under two different settings (high and low) of each of these parameters. Along with the two settings for the shift constraints, this results in a total of $2^4 = 16$ experiments. The high and low values for each parameter are given in Table 1. Green et al. (2001, 2003) additionally studied the effect of the target probability, π. The target level did not appear to have as significant an effect on the reliability of each method as the other parameters did, therefore we do not include different settings of it in our study. Instead of shifts, they included a factor that they call the "planning period" which is similar to the shift factor in that the number of servers is constant in each planning "period" (which can be longer than what we, and they, call a period and measure the performance in).

5.2. Estimating the Pseudogradients and Implementing the SACCPM

Before we describe the implementation of the continuous and discrete optimization problems solved in each iteration of the SACCPM, we discuss what is perhaps the most challenging part of the implementation of the algorithm: estimating the pseudogradients. Up until now, we have assumed that in the implementation of the SACCPM, $\bar{q}^i(\hat{y}; n)$ is a pseudogradient of $\bar{g}_i(\cdot; n)$ at $\hat{y}$. What makes computing (or estimating, because it may prove difficult to ensure that the resulting vector satisfies condition (7) exactly) such a pseudogradient particularly challenging is that the service-level function is a discrete function of the number of agents. In addition, we do not have a closed-form expression of the function.

The simplest and perhaps the most intuitive method for estimating a gradient (or pseudogradient) when an expression for the function is unknown is the method of finite differences (see, e.g., Andradóttir 1998). The method of finite differences can easily be extended to discrete functions, i.e., we let

$$\bar{q}^i_j(\hat{y}; n) = \bar{g}_i(\hat{y} + e_j; n) - \bar{g}_i(\hat{y}; n) \quad \forall j = 1, \ldots, p, \quad (13)$$

where e_j is the jth unit vector in $\mathbb{R}^p$, be the estimate of the pseudogradient of $\bar{g}_i(y; n)$ at the staffing level $\bar{y}$. It is clear that $p + 1$ function evaluations of $\bar{g}_i$ are required to compute $\bar{q}^i$. We note, however, that estimates of the pseudogradients of the service-level functions $\bar{g}_i$ in all p periods, $i = 1, \ldots, p$, can be obtained from those same $p + 1$ simulations.

The number of periods p can be quite large in staffing applications. Therefore, if we could estimate a pseudogradient from a single simulation, the computation time of the SACCPM would be greatly reduced. Atlason (2004) includes, however, an implementation of the SACCPM using infinitesimal perturbation analysis to estimate pseudogradients, and the solutions from that algorithm are inferior to solutions that the algorithm gives when finite differences are used. Pseudogradient estimates based on the likelihood ratio gradient estimation method have also not been reliable (Atlason 2004, Atlason et al. 2003).

At low staffing levels, there are so few servers, and calls are so backed up, that no calls are answered on time, and adding a server does little to alleviate the situation; cf. low values of y_1 and y_2 in Figure 2. It is certainly possible that we would encounter such a staffing level in the cutting-plane algorithm for the sample average approximation of the service-level function; an attempt to compute or estimate a pseudogradient at this point would produce a zero vector. Therefore, as a feasibility cut at such a point $\hat{y}$, we impose a lower bound on the number of servers in the period i, i.e., add the constraint $y_i \geq \hat{y}_i + 1$. This is not necessarily a valid feasibility cut because the reason for calls backing up might be understaffing in previous periods. In our implementation, if such a cut is added during the algorithm, we verify at termination

that the corresponding constraint is not tight at the optimal solution found. If it is tight, then we lower the bound and do more iterations of the cutting-plane method.

Other parts of the SACCPM were implemented as follows. We used a sample size of $n = 100$. We built a simulation model using the ProModel simulation software to compute the sample average of the service-level function. We used Visual Basic for Applications and Microsoft Excel to store the data and to compute the cuts. We used the AMPL modeling language to model, and call a solver for, the analytic center problem (9) and the IP (10) of finding an integer point close to the analytic center. We used the MINOS solver to solve the analytic center problem (9). We used the CPLEX solver to solve the IP (10). We used the Excel solver to compute the cost $f(\hat{y})$ of a particular staffing level $\hat{y}$.

Finally, we describe the settings of the parameters that are specific to the SACCPM. We let $w^k = r$, where r is the number of feasibility cuts that have been added in iterations 1 through $k-1$ (we let $w^k = 1$ if no feasibility cuts have been added). We used $\epsilon = 10^{-5}$ and chose $a = 1 - \epsilon$. Instead of specifying a particular feasible starting point y^0, we started with an upper bound on the staffing levels in each period to ensure boundedness of the localization set. For the parameters chosen in the experiment, it is unlikely that more than 130 agents will be allocated in any period in an optimal solution. Hence, we added the term $-\sum_{i=1}^{p} \log(130 - y_i)$ to the objective of the analytic center problem (9) and the constraints $y_i \le 130$ $\forall i = 1, \ldots, p$ to the IP (10).

5.3. Analytical Queueing Methods

In this section, we describe the queueing methods in Green et al. (2001, 2003) that we use as benchmarks for our computational study. Traditional queueing methods for staffing call centers assume that the process is in steady state, i.e., the arrival rate is fixed and the call center has been open long enough, so that the initial state of the call center does not matter. Then, assuming that the call center can be modeled as an $M/M/s$ queueing system, the probability that a customer must wait more than τ time units, as a function of the number of servers s, is given by (Cooper 1981, pp. 91, 97)

$$P(s) = C(s, R)e^{-(s\mu - \lambda)\tau} \quad \text{for } s > R, \tag{14}$$

where $R = \lambda/\mu$ is the offered load. If $s \le R$, then $P(s) = 1$. The term $C(s, R)$ is the probability that an incoming call is delayed, also called the "Erlang-C probability of delay." It can be computed in a numerically stable way via the recursion (Cooper 1981, pp. 82, 92)

$$B(0, R) = 1,$$

$$B(s, R) = \frac{R \cdot B(s-1, R)}{s + R \cdot B(s-1, R)} \quad \text{for } s \ge 1, \quad \text{and}$$

$$C(s, R) = \frac{sB(s, R)}{R \cdot B(s, R) + s - R} \quad \text{for } s > R.$$

When the arrival rate changes between periods or within periods, Green et al. (2001) proposed adjusting the value of the arrival rate λ used in (14). This is a straightforward method that can give good staffing levels, at least for a call center operation that is similar to the $M(t)/M/s(t)$ model. They consider six different adjustments of the arrival rate. The six different schemes are given in Table 2. In Green et al. (2003), only SIPPavg, LAGavg, and LAGmax are considered. When we computed the arrival rate to use for the LAG methods in the first period, we assumed that the arrival rate prior to time zero was equal to the arrival rate at the beginning of the first period.

The required staffing, y_i, in period i, is computed by letting $\lambda = \Lambda_i$ in (14), and letting

$$y_i = \min\{s > 0 \text{ and integer: } P(s) \le 1 - \pi\}.$$

The cost of the resulting staffing level is $f(y)$.

5.4. Results

The parameter settings and the cost of the staffing levels obtained by each method in each of the 16 experiments are shown in Table 3. The table also shows the cost savings of the SACCPM versus the LAGavg method. We chose the LAGavg method for comparison because it performs best of the six analytical methods. The comparison only includes the feasible solution determined by the criteria described below.

Determining feasibility. The solutions obtained by the six queueing methods and the SACCPM are not guaranteed to be feasible for the call center staffing

Table 2 Different Methods for Adjusting the Arrival Rate to Use in Equation (14)

Method	Λ_i
SIPPavg	$\frac{1}{t_i - t_{i-1}} \int_{t_{i-1}}^{t_i} \lambda(t)\, dt$
SIPPmax	$\max_{t_{i-1} < t \le t_i} \lambda(t)$
SIPPmix	If $\lambda(t)$ is nondecreasing in period i, use SIPPavg rate, otherwise use SIPPmax rate.
LAGavg	$\frac{1}{t_i - t_{i-1}} \int_{t_{i-1} - 1/\mu}^{t_i - 1/\mu} \lambda(t)\, dt$
LAGmax	$\max_{t_{i-1} - 1/\mu < t \le t_i - 1/\mu} \lambda(t)$
LAGmix	If $\lambda(t)$ is nondecreasing in $[t_{i-1} - 1/\mu, t_i - 1/\mu]$, use LAGavg rate, otherwise use LAGmax rate.

Note. Here, t_i is the time when period i ends ($t_0 \equiv 0$), $1/\mu$ is the mean service time, and Λ_i is the rate to be used to determine the staffing in period i.

Table 3 Cost of the Solutions in Agent-Periods

Experiment	μ	R	θ	Shifts	λ	SACCPM	SIPPavg	SIPPmax	SIPPmix	LAGavg	LAGmax	LAGmix	Savings (%)
1	4	8	0.75	Y	32	**1,008**	1,056	1,056	1,056	1,056	1,056	1,056	4.6
2	16	8	0.75	Y	128	**1,032**	1,056	1,056	1,056	**1,032**	1,056	**1,032**	0.0
3	4	32	0.75	Y	128	**3,456**	3,552	3,624	3,576	**3,456**	3,552	3,552	0.0
4	16	32	0.75	Y	512	**3,504**	3,552	3,624	3,576	3,576	3,576	3,528	2.0
5	4	8	0.25	Y	32	**936**	**936**	**936**	**936**	**936**	**936**	**936**	0.0
6	16	8	0.25	Y	128	**936**	**936**	**936**	**936**	**936**	**936**	**936**	0.0
7	4	32	0.25	Y	128	**3,024**	3,048	3,096	3,072	3,048	3,048	3,048	0.8
8	16	32	0.25	Y	512	**2,976**	3,048	3,096	3,072	3,024	3,072	3,048	1.6
9	4	8	0.75	N	32	829	848	862	855	**848**	862	855	—
10	16	8	0.75	N	128	**838**	848	858	853	847	862	853	1.1
11	4	32	0.75	N	128	2,746	2,786	2,838	2,812	2,787	2,838	**2,813**	—
12	16	32	0.75	N	512	2,786	2,786	2,838	2,812	**2,778**	2,830	2,804	−0.3
13	4	8	0.25	N	32	**846**	856	862	859	856	862	859	1.2
14	16	8	0.25	N	128	**850**	854	860	857	856	861	859	0.7
15	4	32	0.25	N	128	**2,774**	2,802	2,818	2,810	2,802	2,818	2,810	1.0
16	16	32	0.25	N	512	**2,790**	2,802	2,818	2,810	2,797	2,815	2,806	0.3

Notes. The bold numbers in each row are the lowest-cost solutions out of the solutions that satisfy the feasibility requirements; see Table 4. "Savings" shows the cost savings of using the SACCPM versus the LAGavg method.

problem with expected service-level constraints (2) (although it is true that a solution obtained by the SACCPM is feasible for the respective sample average approximation problem). To determine feasibility, we increased the sample size to $n = 999$ (the maximum in ProModel) and simulated the call center using each of the solutions found, at the staffing level vector Ax. We simulated at Ax rather than y because the ith component of the vector Ax is the actual number of agents available in period i and can be greater than y_i because of slack in the shift constraint $Ax \geq y$ in (1). Because the simulation output of the 999 experiments is still random, we decided to declare the service-level requirements not met if the fraction of calls answered immediately is less than 75% in any period instead of the $\pi = 80\%$ from the example and was used in the implementation of the SACCPM algorithm. This is similar to what Green et al. (2001) used to determine feasibility. They computed the service level using numerical integration of transient queueing formulas and said that the service level is infeasible in a period if the target probability is exceeded by at least 10%. We chose 75% as our threshold because the largest 95% confidence interval half-width in our simulations using $n = 999$ was 4.4%, and $80\% - 4.4\% \approx 75\%$.

The feasibility of the solutions is reported in Table 4. We first note that all the methods do well in most cases. The SIPPavg method struggles when there are no shifts present and the service rates are low. In this case, there is significant dependence between time periods, which none of the SIPP methods take into account. We also note that the solutions from the SACCPM do not always meet the service-level requirements by the 75% criterion, and in many periods fail to meet the 80% requirements. This is because the sample size was only 100 when the solution was computed, therefore there Is a significant error in the estimates of the service-level functions.

Ranking the methods. In Table 3, we put in boldface text the cost of the method that we declared a "winner." We selected a winner based on two criteria. The first criterion is that the service level must be greater than 75% in all periods based on the simulation using sample size 999. The second criterion is cost. We see that the SACCPM is a winner in all but three experiments. In two of these, the solutions fail to meet the feasibility criterion. In the third case, the cost of the best solution was only 0.3% lower than that obtained by SACCPM. In the case of ties, it would make more sense to use one of the analytical queueing heuristics because they are much easier to implement. However, one would not know beforehand whether a tie would occur, and which heuristic to use.

The analytical queueing heuristics have been shown to do well on this particular type of queueing model. In the SACCPM, however, no assumptions are made on the arrival process or the service times, therefore it may apply in even more general settings.

In §5.2, we noted that the finite difference estimator of a pseudogradient for low staffing levels can be zero. In one or more iterations in some of the experiments, the finite difference method produced a zero vector, which could not be used to generate feasibility cuts. Instead, we imposed the lower bounds described in §5.2 and then checked upon termination of the algorithm whether these lower bounds were tight. The bounds were tight in experiment 16, therefore we relaxed the bounds and ran the algorithm until it terminated again, this time with a solution where these kinds of bounds were not tight.

Table 4 Feasibility of the Solutions and Number of Iterations of the SACCPM

Experiment	SACCPM	SIPPavg	SIPPmax	SIPPmix	LAGavg	LAGmax	LAGmix	Iterations	Feas. cuts	Opt. cuts
1	2 79.0%	1 79.0%	1 79.0%	2 79.0%	81.3%	84.4%	85.9%	17	23	6
2	82.1%	83.1%	83.1%	83.1%	82.1%	83.1%	82.1%	20	36	9
3	1 76.7%	4 **(1)** 74.4%	81.6%	81.5%	82.1%	85.9%	82.5%	10	77	3
4	82.1%	82.0%	84.3%	83.1%	82.0%	86.2%	82.1%	15	97	4
5	82.4%	83.1%	81.3%	82.5%	81.8%	83.3%	83.1%	39	34	16
6	81.8%	81.8%	81.8%	81.8%	81.8%	81.8%	81.8%	39	115	14
7	80.1%	80.4%	82.5%	80.8%	83.5%	83.8%	83.8%	12	123	3
8	3 76.6%	82.0%	82.5%	83.0%	81.6%	82.3%	81.6%	11	119	3
9	15 **(3)** 74.2%	14 **(3)** 73.2%	3 76.2%	3 76.2%	2 76.5%	81.2%	1 79.8%	47	210	6
10	12 77.8%	3 78.4%	80.8%	80.8%	2 79.9%	81.1%	2 79.9%	24	173	7
11	21 **(3)** 74.7%	33 **(23)** 61.2%	29 **(4)** 71.5%	29 **(4)** 71.5%	4 78.7%	80.8%	80.7%	38	374	4
12	5 77.9%	11 76.0%	80.8%	80.8%	1 79.7%	81.9%	80.2%	26	271	5
13	7 76.1%	1 79.9%	80.5%	80.5%	1 80.0%	1 80.0%	1 80.0%	51	273	8
14	6 75.9%	3 79.0%	1 79.8%	2 79.6%	2 79.6%	1 79.8%	2 79.6%	27	290	7
15	17 75.1%	19 76.7%	8 78.2%	8 78.2%	3 79.0%	80.3%	2 79.0%	32	294	4
16	10 76.6%	2 79.1%	81.1%	81.1%	1 79.7%	81.1%	81.1%	53	376	7

Notes. Each cell has up to three values. The first value is the number of periods in which the fraction of calls answered immediately is less than 80%. The boldface values in parentheses are the number of periods in which the fraction of calls answered immediately is less than 75%. The percentages show the lowest fraction of calls answered immediately in any period. We do not display the first two values when they are equal to zero. The last three columns show the number of iterations of the SACCPM and the number of feasibility and optimality cuts added.

5.5. Computational Requirements

We can divide each iteration of the algorithm into three main parts in terms of computations:

1. *Solve the analytic center problem* (9): This usually took less than one second using the MINOS solver and never took more than three seconds.

2. *Solve the IP* (10) *to get an integer solution close to the analytic center*: In the initial stages of the algorithm, this takes on the order of seconds to solve. However, when there were no shifts, meaning that the solution space for y is larger, it could take millions of branch-and-bound nodes to find an optimal solution after a number of cuts were added. Because it is not necessary to find an optimal solution of the IP (10) to advance the algorithm, we often terminated the IP with a suboptimal, but feasible, solution to determine the next iterate. If the IP seemed infeasible, but the infeasibility was difficult to prove and the optimality gap in the SACCPM was less than 1%, the SACCPM was terminated with a possibly suboptimal solution.

Solving the IPs was much easier when the shift constraints were included, which is the scenario one usually faces in practice. In that case, an optimal solution of the IP was found, or the IP was deemed infeasible, in a matter of seconds.

3. *Simulate to estimate the expected service level and gradients*: Simulating at each staffing level in ProModel took from six seconds to about one minute, depending on the number of arrivals to the system, on a Pentium 4 3 GHz computer. Up to 73 such simulations are required per iteration, although we did not always compute all 72 differences to compute the FD pseudogradient. It appeared that dependence between time periods was not a factor over more than 10 periods, so as a rule of thumb, we only computed the differences for the 10 periods preceding period i, and for period i if the service-level constraint was

not satisfied in period i. Staffing levels in subsequent periods do not have any effect on the service level in period i because the requirement is to answer 80% of the calls immediately.

Hence, estimating the pseudogradients and solving the IP (10) requires the most computation; see §6 for ideas on how the computational requirements can be reduced.

The number of iterations required is given in Table 4. In the initial stages, the SACCPM sometimes produced several optimality cuts before any feasibility cuts were generated. Because the weight on the optimality cuts is only one in the beginning, the optimality cuts will be fairly close to each other in such a case, and progress will be slow. This happens when the starting point has much higher staffing levels than what is really needed. When this occurred, we added deeper optimality cuts until the first feasibility cuts were generated by the algorithm, i.e., we lowered z. One could start the algorithm close to the solutions of the analytical queueing heuristics, to hopefully speed up the convergence. We did not try to implement this approach.

6. Conclusions and Future Research

The SACCPM presented in this paper is a promising method for computing optimal staffing levels in a call center when traditional methods fail. Although the computational requirements of our method are large, we were able to solve, or at least approximately solve, a number of moderately sized hypothetical call center staffing problems. The results of the SACCPM were encouraging and they show that the method can potentially be applied in real-world situations with good results. Furthermore, the algorithm can be automated, so that no user intervention is required while the algorithm is running.

The algorithm is robust in the sense that it works well on a range of different problems which gives it further credibility over the traditional methods. Compared to the traditional queueing methods, the SACCPM does best in the presence of shift constraints. To see why, recall that the SACCPM solves both the problem of determining minimum staffing levels and the problem of computing the best assignments to cover these staffing levels simultaneously, while the queueing methods compute the required staffing levels first and the shift assignments afterwards, using the staffing levels as input.

There are several interesting directions for related future research, some of which we are actively pursuing. From the numerical experiments, we identified that both the simulations and solving the integer programs can be computationally expensive. It would be beneficial to study more efficient methods for estimating the pseudogradients that could mimic the performance of the finite difference method. In relation to the integer programs, one should investigate integer programming algorithms that can utilize the special structure of the relaxed problems solved in each iteration and consider allowing approximate solutions of the IPs, especially in early stages of the algorithms. Another technique often used in the continuous case to speed up the computation of the next iterate is to drop cuts that are redundant (see, e.g., Mitchell 2003 for more on this approach), and it is quite possible that the same technique could reduce the time required to solve the IPs in the later stages of the SACCPM. One could also generate some initial cuts by running simulations at the staffing levels suggested by heuristics.

It is conceivable that other optimization methods could perform well in this setting. The extended cutting-plane method in Westerlund and Pörn (2002) seems to fit the framework particularly well, although some details of the implementation are unclear.

The problems solved in this paper were fairly simple instances of a call center staffing problem, but because no assumptions are made on the arrival and service processes and simulation is used to evaluate performance, it seems that the method would also apply in more complicated settings. Call abandonments, skill-based routing, and prioritizing multiple customer classes are problems that call center managers commonly face, and it would be interesting to incorporate those in the algorithm. Moreover, it is likely that the SACCPM could, with appropriate modifications, be applied to problems other than call center staffing.

Acknowledgments

This research was supported by National Science Foundation Grants DMI 0230528 and DMI 0400287 and a Horace H. Rackham School of Graduate Studies Faculty grant. The first author thanks the Department of Industrial and Operations Engineering at the University of Michigan for its generous financial support. The authors thank the two referees for their comments, which lead to improvements in this paper.

References

Andradóttir, S. 1998. Simulation optimization. J. Banks, ed. *Handbook of Simulation*, Chapter 9. John Wiley & Sons, New York, 307–333.

Atkinson, D. S., P. M. Vaidya. 1995. A cutting plane algorithm for convex programming that uses analytic centers. *Math. Programming, Ser. B* 69(1) 1–43.

Atlason, J. 2004. A simulation based cutting plane method for optimization of service systems. Ph.D. thesis, University of Michigan, Ann Arbor, MI.

Atlason, J., M. A. Epelman, S. G. Henderson. 2003. Using simulation to approximate subgradients of convex performance measures in service systems. S. Chick, P. J. Sánchez, D. Ferrin,

D. J. Morrice, eds. *Proc. 2003 Winter Simulation Conf.*, IEEE, Piscataway, NJ, 1824–1832.

Atlason, J., M. A. Epelman, S. G. Henderson. 2004. Call center staffing with simulation and cutting plane methods. *Ann. Oper. Res.* **127** 333–358.

Bahn, O., O. du Merle, J.-L. Goffin, J.-P. Vial. 1995. A cutting plane method from analytic centers for stochastic programming. *Math. Programming, Ser. B* **69**(1) 45–73.

Bazaraa, M. S., H. D. Sherali, C. M. Shetty. 1993. *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons, New York.

Borst, S., A. Mandelbaum, M. I. Reiman. 2004. Dimensioning large call centers. *Oper. Res.* **52**(1) 17–34.

Castillo, I., T. Joro, Y. Li. 2007. Workforce scheduling with multiple objectives. *Eur. J. Oper. Res.* Forthcoming.

Cleveland, B., J. Mayben. 1997. *Call Center Management on Fast Forward*. Call Center Press, Annapolis, MD.

Cooper, R. B. 1981. *Introduction to Queueing Theory*, 2nd ed. Elsevier, North-Holland, New York.

du Merle, O. 1995. Interior points and cutting planes: Development and implementation of methods for convex optimization and large scale structured linear programming. (In French). Ph.D. thesis, University of Geneva, Geneva, Switzerland.

du Merle, O., J.-L. Goffin, J.-P. Vial. 1998. On improvements to the analytic center cutting plane method. *Computational Optim. Appl.* **11** 37–52.

Elhedhli, S., J.-L. Goffin. 2003. The integration of an interior-point cutting plane method within a branch-and-price algorithm. *Math. Programming, Ser. A* **100** 267–294.

Gans, N., G. Koole, A. Mandelbaum. 2003. Telephone call centers: Tutorial, review and research prospects. *Manufacturing Service Oper. Management* **5**(2) 79–141.

Goffin, J.-L., A. Haurie, J.-P. Vial. 1992. Decomposition and nondifferentiable optimization with the projective algorithm. *Management Sci.* **38**(2) 284–302.

Goffin, J.-L., Z.-Q. Luo, Y. Ye. 1996. Complexity analysis of an interior cutting plane method for convex feasibility problems. *SIAM J. Optim.* **6**(3) 638–652.

Green, L. V., P. J. Kolesar, J. Soares. 2001. Improving the SIPP approach for staffing service systems that have cyclic demands. *Oper. Res.* **49**(4) 549–564.

Green, L. V., P. J. Kolesar, J. Soares. 2003. An improved heuristic for staffing telephone call centers with limited operating hours. *Production Oper. Management* **12**(1) 46–61.

Ingolfsson, A., E. Cabral, X. Wu. 2007. Combining integer programming and the randomization method to schedule employees. Working paper, University of Alberta, Alberta, Canada. Retrieved November 28, 2007, http://www.business.ualberta.ca/aingolfsson/publications.htm.

Ingolfsson, A., M. A. Haque, A. Umnikov. 2002. Accounting for time-varying queueing effects in workforce scheduling. *Eur. J. Oper. Res.* **139** 585–597.

Jennings, O. B., A. Mandelbaum, W. A. Massey, W. Whitt. 1996. Server staffing to meet time-varying demand. *Management Sci.* **42**(10) 1383–1394.

Kelley, J. E., Jr. 1960. The cutting-plane method for solving convex programs. *J. Soc. Indust. Appl. Math.* **8**(4) 703–712.

Kleywegt, A. J., A. Shapiro, T. Homem-de-Mello. 2001. The sample average approximation method for stochastic discrete optimization. *SIAM J. Optim.* **12**(2) 479–502.

Kolesar, P. J., L. V. Green. 1998. Insights on service system design from a normal approximation to Erlang's delay formula. *Production Oper. Management* **7** 282–293.

Mandelbaum, A. 2003. Call centers (centres): Research bibliography with abstracts. Version 5. Retrieved June 7, 2004, http://ie.technion.ac.il/serveng/References/ccbib.pdf.

Mason, A. J., D. M. Ryan, D. M. Panton. 1998. Integrated simulation, heuristic and optimisation approaches to staff scheduling. *Oper. Res.* **46**(2) 161–175.

Mitchell, J. E. 2000. Computational experience with an interior point cutting plane algorithm. *SIAM J. Optim.* **10**(4) 1212–1227.

Mitchell, J. E. 2003. Polynomial interior point cutting plane methods. *Optim. Methods and Software* **18**(5) 507–534.

Murota, K. 2003. *Discrete Convex Analysis*. SIAM, Philadelphia.

Nesterov, Y. 1995. Complexity estimates of some cutting plane methods based on the analytic barrier. *Math. Programming, Ser. B* **69**(1) 149–176.

Peton, O., J.-P. Vial. 2001. A tutorial on ACCPM: User's guide for version 2.01. Working paper, University of Geneva, Geneva, Switzerland. Retrieved November 23, 2007, http://blogs.unige.ch/hec/logilab/.

Shapiro, A. 2003. Monte Carlo sampling methods. A. Ruszczynski, A. Shapiro, eds. *Stochastic Programming. Handbooks in Operations Research and Management Science*. Elsevier, Amsterdam, 353–425.

Westerlund, T., R. Pörn. 2002. Solving pseudo-convex mixed integer optimization problems by cutting plane techniques. *Optim. Engrg.* **3** 253–280.

Whitt, W. 1991. The pointwise stationary approximation for $M_t/M_t/s$ queues is asymptotically correct as the rates increase. *Management Sci.* **37**(3) 307–314.

Annals of Operations Research 127, 333–358, 2004

Call Center Staffing with Simulation and Cutting Plane Methods

JÚLÍUS ATLASON * and MARINA A. EPELMAN {jatlason, mepelman}@umich.edu
Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109-2117, USA

SHANE G. HENDERSON shane@orie.cornell.edu
School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY 14853, USA

Abstract. We present an iterative cutting plane method for minimizing staffing costs in a service system subject to satisfying acceptable service level requirements over multiple time periods. We assume that the service level cannot be easily computed, and instead is evaluated using simulation. The simulation uses the method of common random numbers, so that the same sequence of random phenomena is observed when evaluating different staffing plans. In other words, we solve a sample average approximation problem. We establish convergence of the cutting plane method on a given sample average approximation. We also establish both convergence, and the rate of convergence, of the solutions to the sample average approximation to solutions of the original problem as the sample size increases. The cutting plane method relies on the service level functions being concave in the number of servers. We show how to verify this requirement as our algorithm proceeds. A numerical example showcases the properties of our method, and sheds light on when the concavity requirement can be expected to hold.

Keywords: simulation optimization, call centers, cutting planes, sample average approximation

1. Introduction

In this paper we present the theoretical properties of a cutting plane method for minimizing staffing costs in a service system subject to satisfying acceptable service level requirements over multiple time periods. This method was proposed by Henderson and Mason (1998) and combines simulation and integer programming in an iterative cutting plane algorithm. Simulation is a powerful method for analyzing complex systems, but optimization with simulation can be difficult. Linear integer programming problems, along with many other mathematical programming models, are well studied and many algorithms have been developed for solving problems in this form, but a simplification of the system is often required for modelling. Our iterative cutting-plane algorithm combines simulation and linear (integer) programming to solve resource allocation problems where the objective function, or constraints, or both, are evaluated via simulation. The algorithm relies on the concavity of the problem constraints, but in our algorithm we have a built-in robustness, so that nonconcavity can be detected.

* Corresponding author.

The method of combining simulation and optimization in this way has potential applications in various service systems, such as call center staffing (which will be the focus of this paper) and emergency vehicle dispatching (which we are currently investigating). In fact, the method could potentially, with appropriate modifications, be utilized in many other areas where simulation is an appropriate modelling tool.

The problem of determining optimal staffing levels in a call center (see, e.g., Thompson (1997)) is a motivating example for our work. The decision maker faces the task of creating a collection of tours (work schedules) for the call center of low cost that together ensure a satisfactory service level. A tour is comprised of several shifts and has to observe several restrictions related to labor contracts, management policies, etc. We divide the planning horizon (typically a day or a week) into small periods (15–60 minutes) and focus on the service level in each period. We define the service level in a given period as the fraction of calls received in that period answered within a specified time limit. In this paper we focus on the problem of minimizing cost while satisfying the service level and scheduling constraints.

The traditional method for solving this problem involves two steps. First, the required staffing level in each period is estimated, independently period by period, often using i.e. queueing theory. Second, an integer program is solved to determine how many workers should be assigned to each of the tours in order to "cover" the previously assigned staffing levels. Our method combines these two steps and allows for dependence between periods. There are examples in the literature (Green, Kolesar, and Soares, 2001; Ingolfsson, Haque, and Umnikov, 2002; Jennings et al., 1996) that show that significant cost savings can be obtained by doing so, or that a staffing level obtained by assuming independence does not meet performance criteria when there is, in fact, dependence between periods. Indeed, we present an example at the end of the paper showing that the staffing level in one period can have a considerable effect on the service level in another period. Green, Kolesar, and Soares (2001) and Jennings et al. (1996) suggest a relatively simple method for determining the required staffing levels that accounts for such dependence. Their method is based on infinite server queuing models, but requires that the call center can be accurately modelled as a $G(t)/G(t)/s(t)$ queuing system.

The cost function is usually relatively straightforward to calculate. We can calculate the cost of each tour (salary costs, appeal to employees, etc.), and multiply by the number of employees working each tour to get the overall cost. The service level, on the other hand, can be very difficult to obtain. Queuing models can be used for simple problems, but simulation must be used to accurately model complex systems. The difficulty with using simulation is the large number of possible solutions since it is impractical to evaluate all of them. By using integer programming, we hope that we only need to simulate a small portion of the solution set.

Simulation has been widely used to analyze the impact of different staffing levels on service levels and commercial simulation packages, specially designed for call centers, are available. Integer programming has also been used, in which case the staff requirements in each period are usually needed as an input in the model (see Mehrotra, Murphy, and Trick (2000)).

We present a cutting plane method based on the one developed by Kelley, Jr. (1960). The method solves a linear (integer) program to obtain the staffing levels, and the solution is used as an input for a simulation to calculate the service level. If the service level is unsatisfactory, we add a constraint to the linear program and go to the next iteration.

Kelley's cutting plane method applies to minimization problems where both the objective function and feasible region (of the continuous relaxation of the integer problem) need to be convex. The costs in the call center problem are linear and we will assume that the service level function is concave, so that (see equation (1)), the feasible region, relaxing the integer restriction, is convex. Since the service level function is unknown beforehand, we need to incorporate a mechanism into the method to verify that the concavity assumption holds. In section 5 we present a numerical method for checking concavity of a function, when the function values and possibly gradients are only known at a finite number of points.

Morito et al. (1999) use simulation in a cutting-plane algorithm to solve a logistic system design problem at the Japanese Postal Service. Their problem is to decide where to sort mail provided that some post offices have automatic sorting machines but an increase in transportation cost and handling is expected when the sorting is more centralized. The algorithm proved to be effective for this particular problem and found an optimal solution in only 3 iterations where the number of possible patterns (where to sort mail for each office) was 2^{30}. Their discussion of the algorithm is ad hoc, and they do not discuss its convergence properties.

Ingolfsson, Haque, and Umnikov (2002) present an algorithm for solving a call center staffing problem that uses a genetic algorithm for the optimization component and numerical solution of differential equations for evaluating the service level. Ingolfsson and Cabral (2002) have developed a cutting plane algorithm for this problem using queuing models instead of simulation to calculate the service levels. The cuts are generated using a heuristic, based on approximating the service level in each period as a function of the staffing level in that period, and may not be valid, although examples suggest good performance.

Cutting plane methods have been successfully used to solve two stage stochastic linear programs. In many applications the sample space becomes so large that one must revert to sampling to get a solution (Birge and Louveaux, 1997; Infanger, 1994). The general cutting plane algorithm for two stage stochastic programming is known as the L-shaped method (van Slyke and Wets, 1969) and is based on Benders decomposition (Benders, 1962). Stochastic decomposition (Higle and Sen, 1991) for solving the two stage stochastic linear program starts with a small sample size, which is increased as the algorithm progresses and gets closer to a good solution. Stochastic decomposition could also be applied in our setting, but that is not within the scope of this paper.

The random nature of our problem and the absence of an algebraic form for the service level function makes the optimization challenging. We use sampling to get an estimate of the service level function, and optimize the sample average approximation.

An important question is whether the solution to the sample average approximation converges to a solution to the original problem, and if so, how fast.

We apply the strong law of large numbers to prove conditions for almost sure convergence and apply a result due to Dai, Chen, and Birge (2000) to prove an exponential rate of convergence of the optimal solutions as the sample size increases. Vogel (1994) proved almost sure convergence in a similar setting, but we include proofs for reasons listed in section 4.1. Shapiro and Homem-de-Mello (2000) established conditions for an exponential rate of convergence of the probability that the solution to the sample average approximation is exactly the solution to the original problem in the case of a discrete distribution and Vogel (1988) proved a polynomial rate of convergence in a similar setting, but under weaker conditions than we require. The optimization of sample average approximations has also been studied in the simulation context (Chen and Schmeiser, 2001; Healy and Schruben, 1991; Robinson, 1996; Rubenstein and Shapiro, 1993).

The main contribution of this paper is to demonstrate the potential of bringing simulation and traditional optimization methods together. We establish the properties of a new method for solving call center staffing problems. The method is carefully developed because we believe that the same idea can be applied to resource allocation problems other than staffing problems, as previously mentioned. In addition, we present a numerical method for checking the concavity of a function when the function value and possibly gradient is only known at a finite number of points.

The computing requirements of the algorithm presented here, as applied to realistically-sized problems, are rather large. Indeed, it is often the case that the covering integer programs alluded to earlier (with predetermined staffing levels in each period) are difficult to solve, so that iterating such a step with simulation appears to be a rather formidable computational task. We view this work as a first step in the process of integrating the steps of determining work requirements and covering the work requirements with tours. Subsequent work will focus on exploring methods for making the approach computationally feasible. We have many ideas for how this could be achieved; see section 7 for more comments on this issue.

The paper is organized as follows. We formulate the call center staffing problem in section 2. We present the cutting plane algorithm and its convergence properties in section 3. The convergence and the rate of convergence of the solutions of the sample average approximation to solutions of the original problem are proved in section 4. The numerical method for checking concavity is described in section 5 and an implementation of the overall method is described in section 6. Conclusions and considerations for future research are given in section 7.

2. Call center staffing

In this section we formulate and discuss in more detail the call center staffing problem of minimizing cost subject to service level constraints.

2.1. *Formulation and notation*

The management of a call center needs some criteria to follow when they decide on a set of staffing levels. It is not unusual in practice to determine the staffing levels from a service level perspective. In an emergency call center, for example, it might be required that 90% of received calls should be answered within 10 seconds.

We introduce terminology and notation before we formulate the problem. The set of permissible tours (predefined work schedules over the planning horizon) can be conveniently set up in a matrix (see Dantzig (1954)). More specifically we have

$$A_{ij} = \begin{cases} 1, & \text{if period } i \text{ is included in tour } j, \\ 0, & \text{otherwise.} \end{cases}$$

From the above we see that a column in A represents a feasible tour and a row in A represents a specific period. We let p be the total number of periods and m be the number of feasible tours. If we let $x \in \mathbb{R}^m$ be a vector where the jth component represents the number of employees that work tour j, then $Ax = y \in \mathbb{R}^p$ is a vector where the ith component of y corresponds to the number of employees that are working in period i. We let c be the cost vector, where c^j is the cost per employee working tour j.

Next we define the service level constraints. We let $l \in \mathbb{R}^p$ be the vector whose ith component is the minimum acceptable service level in period i, for example, 90%. Since, for example, the arrival and service times of customers are not known but are random, the service level in each period will be a random variable. Let ξ, a random vector, denote all the random quantities in the problem and let $\xi^1, \ldots, \xi^n$ denote independent realizations of ξ. Let $N^i(\xi)$ be the number of calls received in period i and let $S^i(y, \xi)$ be the number of those calls answered within a pre-specified time limit, for example, 10 seconds, based on the staffing level y. The fraction of customers receiving adequate service in period i in the long run is then

$$\lim_{n\to\infty} \frac{\sum_{d=1}^n S^i(y, \xi^d)}{\sum_{d=1}^n N^i(\xi^d)} = \frac{\lim_{n\to\infty} n^{-1} \sum_{d=1}^n S^i(y, \xi^d)}{\lim_{n\to\infty} n^{-1} \sum_{d=1}^n N^i(\xi^d)}.$$

If $E[N^i(\xi)] < \infty$ then the strong law of large numbers can be applied separately to both the numerator and denominator of this expression, and then the desired long-run ratio is $E[S^i(y, \xi)]/E[N^i(\xi)]$. Thus, $E[S^i(y, \xi)]/E[N^i(\xi)] \geqslant l^i$ is a natural representation of the service level constraint (excluding the pathological case $E[N^i(\xi)] = 0$) in period i. If we define $G^i(y, \xi) := S^i(y, \xi) - l^i N^i(\xi)$ then we can conveniently write the service level constraint as $E[G^i(y, \xi)] \geqslant 0$. Define $g^i(y) := E[G^i(y, \xi)]$ as the expected service level in period i as a function of the server allocation vector y and let $g: \mathbb{R}^p \to \mathbb{R}^p$ be a function whose ith component is g^i.

We are now ready to formulate the problem of minimizing staffing costs subject to satisfying a minimum service level in each period. It is

$$\begin{aligned} \min\ & c^T x \\ \text{subject to}\ & Ax \geqslant y, \\ & g(y) \geqslant 0, \\ & x \in X, \\ & x, y \geqslant 0 \text{ and integer.} \end{aligned} \tag{1}$$

where X is a compact set. The compactness of X can be easily justified in practice. It is, for example, impossible to hire an infinite number of employees, and there are usually budget constraints which impose an upper bound on x since c is generally positive. We also define, for future reference,

$$Y := \{y \geqslant 0 \text{ and integer: } \exists\, 0 \leqslant x \in X \text{ and integer with } Ax \geqslant y\}.$$

Note that Y is a finite set since X is compact and the entries in A are either 0 or 1.

The functions $g^i(y)$ are expected values, and the underlying model might be so complex that an algebraic expression for $g(y)$ can not be easily obtained. Therefore, simulation could be the only viable method for estimating $g(y)$. In the next subsection we formulate (1) as an approximate problem, where the expected values are replaced by sample averages.

2.2. *Sample average approximation of the call center problem*

In this paper we assume that the algebraic form of the service level function $g(y)$ is not available, and that its value is evaluated using simulation. Suppose we run a simulation with sample size n, where we independently generate the realizations $\{\xi^d\}_{d=1}^n$ from the distribution of ξ, to get an estimate of the expected values $g(y)$. Let $\bar{g}_n(y) = (1/n)\sum_{d=1}^n G(y, \xi^d)$ be the resulting estimates and let $\bar{g}_n^i(y)$ denote the ith component of $\bar{g}_n(y)$. We use this notation to formulate the sample average approximation

$$\begin{aligned} \min\ & c^T x \\ \text{subject to}\ & Ax \geqslant y, \\ & \bar{g}_n(y) \geqslant 0, \\ & x \in X, \\ & x, y \geqslant 0 \text{ and integer.} \end{aligned} \tag{2}$$

The problem above is linear except for the service level function $\bar{g}_n(y)$. We assume that each of the component functions $\bar{g}_n^i(y)$ are concave so that we can approximate them with piecewise linear concave functions and solve the sample average approximation by using cutting plane methods. In the next subsection we discuss the concavity assumption in more detail.

2.3. *Concave service levels*

Intuitively, we would expect that the service level increases if we increase the number of employees in any given period. We also conjecture that the marginal increase in service level decreases as we add more employees. If these speculations are true then $g^i(y)$ is increasing and concave in each component of y for all i. We will make the stronger assumption that $g^i(y)$ and $\bar{g}_n^i(y)$ are increasing componentwise and jointly concave in y, for all i. Our initial computational results suggest that this is a reasonable assumption, at least within a region containing practical values of y (see section 6). Others have also studied the convexity of performance measures of queuing systems. Akşin and Harker (2001) show that the throughput of a call center is stochastically increasing and directional concave in the sample path sense as a function of the allocation vector y in a similar setting. Analysis of the steady state waiting time of customers in an $M/M/s$ queue shows that its expected value is a convex and decreasing function of the number of servers s (Dyer and Proll, 1977), its expected value is convex and increasing as a function of the arrival rate (Chen and Henderson, 2001) and its distribution function evaluated at any fixed value, is concave and decreasing as a function of the arrival rate (Chen and Henderson, 2001). See other references in Chen and Henderson (2001) for further studies in this direction. Koole and van der Sluis (2003) developed a local search algorithm for a call center staffing problem with a global service level constraint. When the service level constraint satisfies a property called multimodularity their algorithm is guaranteed to terminate with a global optimal solution. There are, however, examples where the service level constraint, as defined in this paper, is not multimodular even for nondecreasing and concave service level functions.

If the concavity assumption holds, then we can approximate the service level function with piecewise linear concave functions, which can be generated as described below. The following definition is useful.

Definition 1 (Rockafellar, 1970, p. 308). Let $y^k \in \mathbb{R}^p$ be given. If $h: \mathbb{R}^p \to \mathbb{R}$ is a concave function and $q(y^k) \in \mathbb{R}^p$ is such that

$$h(y) \leqslant h(y^k) + q(y^k)^T(y - y^k) \quad \forall y \in \mathbb{R}^p \tag{3}$$

then $q(y^k)$ is a subgradient of h at y^k.

The term "supergradient" might be more appropriate since the hyperplane $\{h(y^k) + q(y^k)^T(y - y^k)\}$ lies above the function h, but we use "subgradient" to conform with the literature. A concave function has at least one subgradient at every point (see theorem 3.5.2 in Bazaraa, Sherali, and Shetty (1993)). The notion of concavity and subgradients is defined for functions of continuous variables, but we are dealing with functions of integer variables. We say that such a function h is concave if no points of the form $(x, h(x)) \in \mathbb{R}^{p+1}$ (with $x \in \mathbb{Z}^p$) lie in the interior of the convex hull of the set $\{(y, h(y)): y \in \mathbb{Z}^p\} \subseteq \mathbb{R}^{p+1}$. We replace $\mathbb{R}^p$ with $\mathbb{Z}^p$ in definition 1 to define the subgradient of a function with integer domain.

Let $q^i(y^k)$ and $\bar{q}_n^i(y^k)$ be subgradients at y^k of g^i and $\bar{g}_n^i$, respectively. There are many potential methods one might consider to obtain the subgradients. Finite differences using differences of length 1 appear reasonable since we are working with integer variables. There are, however, examples where that fails to produce a subgradient, even for a concave nondecreasing function. Still, we used finite differences in our numerical study and converged to an optimal solution of the sample average approximation. Gradients might also be obtained using infinitesimal perturbation analysis (IPA) (see, e.g., Glasserman (1991)). Before using IPA we would have to extend the service level function to a differentiable function defined over a continuous domain, since IPA is applied in settings where the underlying function is differentiable.

The subgradients are used to approximate the service level constraints. Let y^k be a given server allocation vector, and suppose that $\bar{g}_n^i(y^k)$ and $\bar{q}_n^i(y^k)$ are obtained via simulation. If our assumptions about concavity hold then by definition 1 we must have $\bar{g}_n^i(y) \leqslant \bar{g}_n^i(y^k) + \bar{q}_n^i(y^k)^T(y - y^k)$ for all allocation vectors y, and all i. We want y to satisfy $\bar{g}_n(y) \geqslant 0$ and therefore it is necessary that

$$0 \leqslant \bar{g}_n^i(y^k) + \bar{q}_n^i(y^k)^T(y - y^k), \tag{4}$$

for all i.

In the next section we show how to use the subgradients in a cutting plane algorithm to solve the sample average approximation (2).

3. The cutting plane method

In this section we present a cutting plane algorithm for solving the sample average approximation (2). We select a fixed sample size at the beginning of the algorithm and use the same sample (common random numbers) in each iteration. This minimizes the effect of sampling in that we only work with one function $\bar{g}_n$ instead of getting a new $\bar{g}_n$ function in each iteration, which could, for example, invalidate the concavity assumption.

The typical cutting plane algorithm for (2) works as follows. We relax the nonlinear service level constraints to convert the call center staffing problem into a linear integer problem. We solve the linear integer problem and run a simulation with the staffing levels obtained from the solution. If the service levels meet the service level constraints as approximated by the sample average then we stop with an optimal solution to (2). If a service level constraint is violated then we add a linear constraint to the relaxed problem that eliminates the current solution but does not eliminate any feasible solutions to the sample average approximation.

Our algorithm fits the framework of Kelley's cutting plane method (Kelley, Jr., 1960). It differs from the traditional description of the algorithm only in that we use a stimulation to generate the cuts and evaluate the function values instead of having an algebraic form for the function and using analytically determined gradients to generate the cuts. Nevertheless, we include a proof of convergence of our cutting plane method, since its statement is specific to our algorithm and it makes the results clearer.

The relaxed problem for (2) that we solve in each iteration is

$$\begin{aligned} \min\ & c^T x \\ \text{subject to}\ & Ax \geqslant y, \\ & D_k y \geqslant d_k, \\ & x \in X, \\ & x, y \geqslant 0 \text{ and integer.} \end{aligned} \tag{5}$$

We replaced the constraints $\bar{g}_n(y) \geqslant 0$ with linear constraints $D_k y \geqslant d_k$. The subscript k indicates the iteration number in the cutting plane algorithm. The constraint set $D_k y \geqslant d_k$ is initially empty but we add more constraints to it as the algorithm evolves.

At iteration k we solve an instance of (5) to obtain the solution pair (x^k, y^k). For the server allocation vector y^k we run a simulation to calculate $\bar{g}_n(y^k)$. If we find that the service level is unacceptable, i.e., if $\bar{g}_n^i(y^k) < 0$ for some i, then we add the constraint (4) to the set $D_k y \geqslant d_k$, i.e., we add the component $-\bar{g}_n^i(y^k) + \bar{q}_n^i(y^k)^T y^k$ to d_k and the row vector $\bar{q}_n^i(y^k)^T$ to D_k. We add a constraint for all periods i where the service level is unacceptable. Otherwise, if the service level is acceptable in all periods then we terminate the algorithm with an optimal solution to the sample average approximation (2).

Algorithm 1.

Initialization. Generate n independent realizations from the distribution of ξ. Let $k \leftarrow 1$, D_1 and d_1 be empty.
Step 1. Solve (5) and let (x^k, y^k) be an optimal solution.
Step 1a. Stop with an error if (5) was infeasible.
Step 2. Run a simulation to obtain $\bar{g}_n(y^k)$.
Step 2a. If $\bar{g}_n(y^k) \geqslant 0$ then stop. Return (x^k, y^k) as an optimal solution.
Step 3. Compute, by simulation, $\bar{q}_n^i(y^k)$ for all i for which $\bar{g}_n^i(y^k) < 0$, and add the cuts (4) to D_k and d_k.
Step 4. Let $d_{k+1} \leftarrow d_k$ and $D_{k+1} \leftarrow D_k$. Let $k \leftarrow k + 1$. Go to step 1.

It is usually not necessary to store the n independent realizations referred to in the initialization phase. Instead, we only need to store a few numbers, called seeds, and reset the random number generators (streams) in the simulation with the seeds at the beginning of each iteration. See Law and Kelton (2000) for more details on this approach to using common random numbers. To speed up the algorithm it is possible to start with D_1 and d_1 nonempty. Ingolfsson and Cabral (2002) developed, for example, lower bounds on y. They point out that if there is an infinite number of servers in all periods except period i and if $\tilde{y}_i$ is the minimum number of employees required in period i in this setting so that the service level in period i is acceptable, then $y_i \geqslant \tilde{y}_i$ for all y satisfying $g(y) \geqslant 0$. We could select D_1 and d_1 to reflect such lower bounds.

If the algorithm terminates in step 1a then the sample average approximation is infeasible. That could be due to either a sampling error, i.e., the sample average approximation does not have any feasible points even though the original problem is feasible, or that the original problem is infeasible. As a remedy, either the sample size should be

increased, or the original problem should be reformulated, e.g., the acceptable service level should be lowered, or more employees should be allocated (expand X).

In the algorithm above we solve an integer linear program and add constraints to it in each iteration until we terminate. The integer linear problem always has a larger feasible region than the sample average approximation (2), so $c^T x^k \leqslant c^T x^{k+1} \leqslant c^T x_n^*$, where (x_n^*, y_n^*) is an optimal solution for (2). An important question is whether $\lim_{k\to\infty} c^T x^k = c^T x_n^*$. The following theorem answers this question in the positive.

Theorem 1.

1. The algorithm terminates in a finite number of iterations.
2. Suppose that each component of $\bar{g}_n$ is concave in y. Then the algorithm terminates with an optimal solution to (2) if and only if (2) has a feasible solution.

Proof. 1. Y is a finite set and it is therefore sufficient to show that no point in Y is visited more than once. Suppose that the algorithm did not terminate after visiting point y^t. That means that $\bar{g}_n(y^t) \not\geqslant 0$ and we added one or more cuts of the form

$$0 \leqslant \bar{g}_n^i(y^t) + \bar{q}_n^i(y^t)^T(y - y^t), \quad i \in \{1, \ldots, p\}$$

to (5). Suppose that $y^k = y^t$, for some $k > t$. Since y^k is the solution for (5) at step k it must satisfy the cuts added at iteration t, i.e., $0 \leqslant \bar{g}_n^i(y^t) + \bar{q}_n^i(y^t)^T(y^k - y^t) = \bar{g}_n^i(y^t)$, which is a contradiction because this constraint was added since $\bar{g}_n^i(y^t) < 0$. Hence, we visit a new point in the set Y in each iteration and thus the algorithm terminates in a finite number of iterations.

2. Suppose first that (2) does not have a feasible solution. Then no $y \in Y$ satisfies $\bar{g}_n(y) \geqslant 0$. The algorithm only visits points in Y, so the optimality condition in step 2a is never satisfied. Since the algorithm terminates in a finite number of iterations it must terminate with the relaxed problem being infeasible. Suppose now that (2) is feasible. The problem (5) solved in step 1 is a relaxed version of (2) since $\bar{g}_n$ is concave, so (5) is feasible in every iteration. Therefore, the algorithm terminates in step 2a with (x^k, y^k) as the solution. But $\bar{g}_n(y^k) \geqslant 0$ by the termination criteria, so it is an optimal solution to (2). □

The theorem above states that we terminate with an optimal solution to the sample average approximation so long as one exists. In the next section, we discuss the convergence of that solution to an optimal solution to the original problem (1) as the sample size n increases.

4. Convergence of solutions of the sample average approximation to solutions of the original problem

We have established that the cutting plane algorithm will identify an optimal solution of the problem (2). The problem (2) was formed by approximating the expected service level constraints of problem (1), and we will next investigate if solutions of the sample

average approximation converge to a solution of the original problem. We show, by using the strong law of large numbers (SLLN), that the set of optimal solutions of the sample average approximation is a subset of the set of optimal solutions for the original problem w.p. 1 as the sample size gets large. Furthermore, we show that the probability of this event approaches 1 exponentially fast when we increase the sample size. These results require the existence of at least one optimal solution for the original problem to satisfy the expected service level constraints with strict inequality, but this regularity condition can be easily justified for practical purposes as will be discussed later.

4.1. *Almost sure convergence of optimal solutions of the sample average approximation*

The results in this section may be established by specializing the results in Vogel (1994). We choose to provide direct proofs in this section for 3 main reasons:

1. The additional structure in our setting allows a clearer statement and proof of the results.
2. The proofs add important insight into why solving the sample average approximation is a sensible approach.
3. The proofs serve as an excellent foundation to develop an understanding of the "rate of convergence" results that follow in section 4.2.

The effect of the sampling on the optimization problem is to change the shape of the feasible region. It directly affects the service level constraint, so we will rewrite the problems (1) and (2) to make the effect more transparent and to make the proofs easier to read. First define

$$f(y) := \min_{\{x \geqslant 0 \text{ and integer: } x \in X,\ Ax \geqslant y\}} c^T x,$$

where $f(y) = +\infty$ if the set $\{x \geqslant 0$ and integer: $x \in X,\ Ax \geqslant y\}$ is empty. Now we can rewrite problem (1) as

$$\begin{array}{ll} \min & f(y) \\ \text{subject to} & y \in Y, \\ & g(y) \geqslant 0 \end{array} \tag{6}$$

and its sample average approximation, which is equivalent to (2), as

$$\begin{array}{ll} \min & f(y) \\ \text{subject to} & y \in Y, \\ & \bar{g}_n(y) \geqslant 0. \end{array} \tag{7}$$

We are interested in the properties of the optimal solutions of (7) as the sample size n gets large. It turns out, by an application of the SLLN, that any optimal solution of (6) that satisfies $g(y) > 0$, i.e., $g^i(y) > 0$ for all i, is an optimal solution of (7) with

probability 1 (w.p. 1) as n goes to infinity. We make a few more definitions before we prove this. Let

$$\bar{g}_\infty(y) := \lim_{n\to\infty} \bar{g}_n(y),$$
$$F^* := \text{the optimal value of (6)}$$

and define the sets

$$Y^* := \text{the set of optimal solutions to (6)},$$
$$Y_0^* := \{y \in Y^*:\ g(y) > 0\},$$
$$Y_1 := \{y \in Y:\ f(y) \leqslant F^*,\ g(y) \not\geqslant 0\},$$
$$Y_n^* := \text{the set of optimal solutions to (7)}.$$

Note that Y_1 is the set of solutions to (6) that have the same or lower cost than an optimal solution, and satisfy all constraints except the service level constraints. We are concerned with solutions in this set since they could be feasible (optimal) to the sample average approximation (7) if the difference between the sample average, $\bar{g}_n$, and g is sufficiently large. We show that when Y_0^* is not empty, $Y_0^* \subseteq Y_n^* \subseteq Y^*$ for all n large enough w.p. 1. We say that property $E(n)$ holds for all n large enough w.p. 1 if and only if $P[\exists N < \infty:\ E(n) \text{ holds } \forall n \geqslant N] = 1$. (Here N should be viewed as a random variable.) Sometimes such statements are communicated by saying that $E(n)$ holds *eventually*.

We start with two lemmas. The first one establishes properties of $\bar{g}_\infty(y)$ by repeatedly applying the SLLN. The second shows that solutions to (6) satisfying $g(y) > 0$, and infeasible solutions, will be feasible and infeasible, respectively, w.p. 1 for problem (7) when n gets large. The only condition $g(y)$ has to satisfy is that it has to be finite for all $y \in Y$. That assumption is easily justified by noting that the absolute value of each component of $g(y)$ is bounded by the expected number of arrivals in that period, which would invariably be finite in practice.

Even though we restrict attention to optimal solutions, the overall approach would not change if we wanted to prove that all "interior" feasible solutions for (6) are eventually feasible for (7) w.p. 1 and that all infeasible solutions for (6) are eventually infeasible for (7). This may lend some intuition, since it will then almost invariably be the case that the feasible region of the sample average approximation converges to the feasible region of the original problem and therefore the set of optimal solutions converges. Define

$$\|g\| = \max_{y\in Y} \|g(y)\|_\infty = \max_{y\in Y} \max_{i=1,\ldots,p} |g^i(y)|.$$

Lemma 2.

1. Suppose that $\|g(y)\|_\infty < \infty$ for some fixed $y \in \mathbb{Z}^p$, $y \geqslant 0$. Then $\bar{g}_\infty(y) = g(y)$ w.p. 1.
2. Suppose that $\|g\| < \infty$ and $\Gamma \subseteq Y$. Then $\bar{g}_\infty(y) = g(y)\ \forall y \in \Gamma$ w.p. 1.

Proof. 1. The SLLN (see theorem 6.1 in Billingsley (1995)) gives $\bar{g}^i_\infty(y) = g^i(y)$ w.p. 1. So

$$P\big[\bar{g}_\infty(y) = g(y)\big] \geqslant 1 - \sum_{i=1}^{p} P\big[\bar{g}^i_\infty(y) \neq g^i(y)\big] = 1.$$

2. Note that

$$P\big[\bar{g}_\infty(y) = g(y)\ \forall y \in \Gamma\big] \geqslant 1 - \sum_{y\in\Gamma} P\big[\bar{g}_\infty(y) \neq g(y)\big] = 1$$

since Γ is finite. □

Lemma 3. Suppose that $\|g\| < \infty$. Then

1. $\bar{g}_n(y) \geqslant 0\ \forall y \in Y_0^*$ for all n large enough w.p. 1.
2. All $y \in Y_1$ are infeasible for the sample average approximation (7) for all n large enough w.p. 1.

Proof. 1. The result is trivial if Y_0^* is empty, so suppose it is not. Let

$$\epsilon = \min_{y\in Y_0^*} \min_{i\in\{1,\ldots,p\}} \{g^i(y)\}.$$

Then $\epsilon > 0$ by the definition of Y_0^*. Let

$$N_0 = \inf\Big\{n_0\colon \max_{y\in Y_0^*} \|\bar{g}_n(y) - g(y)\|_\infty < \epsilon \quad \forall n \geqslant n_0\Big\},$$

with the infimum defined as $+\infty$ if the set is empty, and then $\bar{g}_n \geqslant 0\ \forall y \in Y_0^*\ \forall n \geqslant N_0$. Now, the set Y_0^* is a subset of Y, so $\lim_{n\to\infty} \bar{g}_n(y) = g(y)\ \forall y \in Y_0^*$ w.p. 1 by part 2 of lemma 2. Therefore, $N_0 < \infty$ w.p. 1.

2. The result is trivial if Y_1 is empty, so suppose it is not. Let

$$\epsilon = \min_{y\in Y_1} \max_{i\in\{1,\ldots,p\}} \{-g^i(y)\}.$$

Then $\epsilon > 0$, since $g^i(y) < 0$, for at least one $i \in \{1, \ldots, p\}\ \forall y \in Y_1$. Let

$$N_1 = \inf\{n_1\colon \|g(y) - \bar{g}_n(y)\|_\infty < \epsilon \quad \forall n \geqslant n_1\}$$

and then all $y \in Y_1$ are infeasible for (7) for all $n \geqslant N_1$. Now, the set Y_1 is a subset of Y, so $\lim_{n\to\infty} \bar{g}_n(y) = g(y)\ \forall y \in Y_1$ w.p. 1 by part 2 of lemma 2. Therefore, $N_1 < \infty$ w.p. 1. □

Lemma 3 shows that all the "interior" optimal solutions for the original problem are eventually feasible for the sample average approximation and remain so as the sample size increases. Furthermore, all solutions that satisfy the constraints that are common for both problems, but not the service level constraints, and have at most the same cost as an optimal solution, eventually become infeasible for the sample average approximation. Hence, we have the important result that for a large enough sample size an optimal solution for the sample average approximation is indeed optimal for the original problem.

Theorem 4. Suppose that $\|g\| < \infty$. Then $Y_0^* \subseteq Y_n^*$ for all n large enough w.p. 1. Furthermore, if Y_0^* is nonempty then $Y_0^* \subseteq Y_n^* \subseteq Y^*$ for all n large enough w.p. 1.

Proof. The first inclusion holds trivially if Y_0^* is empty, so assume that Y_0^* is not empty. On each sample path let $N = \sup\{N_0, N_1\}$, where N_0 and N_1 are the same as in lemma 3. When $n \geq N$ we know that all $y \in Y_0^*$ are feasible for (7) and that all $y \in Y_1$ are infeasible for (7). Hence, all $y \in Y_0^*$ are optimal for (7) and no $y \notin Y^*$ is optimal for (7) whenever $n \geq N$. Thus, $Y_0^* \subseteq Y_n^* \subseteq Y^*$ for all $n \geq N$. Finally, $P[N < \infty] = P[N_0 < \infty, N_1 < \infty] \geq P[N_0 < \infty] + P[N_1 < \infty] - 1 = 1$. □

Corollary 5. Suppose that $\|g\| < \infty$ and that (1) has a unique optimal solution, y^*, such that $g(y^*) > 0$. Then y^* is the unique optimal solution for (2) for all n large enough w.p. 1.

Proof. In this case $Y_0^* = Y^* = \{y^*\}$ and the result follows from the previous theorem. □

The conclusion of theorem 4 relies on existence of an "interior" optimal solution for the original problem. A simple example illustrates how the conclusion can fail if this requirement is not satisfied. Let ξ be a uniform random variable on $[-0.5, 0.5]$ and consider the following problem:

$$\begin{aligned} &\min\ y \\ &\text{subject to}\quad y \geq \left|E[\xi]\right|, \\ &\qquad\qquad y \geq 0 \text{ and integer.} \end{aligned}$$

Then $y^* = 0$ for this problem since $E[\xi] = 0$. We form the sample average approximation by replacing $E[\xi]$ with $\bar{\xi}_n$, the sample average of n independent realizations of ξ. Then $0.5 > |\bar{\xi}_n| > 0$ w.p. 1 for all $n > 0$ and thus we get that $y_n^* = 1$ w.p. 1.

We mentioned earlier that the existence of an "interior" optimal solution is merely a regularity condition. In reality it is basically impossible to satisfy the service level constraints in any period exactly, since the feasible region is discrete. Even if this occurred, we could subtract an arbitrarily small positive number, say ε, from the right-hand side of each service level constraint and solve the resulting ε-perturbed problem. Then all solutions with $g^i(y) = 0$ for some i satisfy $g^i(y) > -\varepsilon$ and it is sufficient for the problem to have an optimal solution (not necessarily satisfying $g(y) > 0$) for theorem 4 to hold. This rationale also applies to the next subsection where we prove an exponential rate of convergence as the sample size increases.

4.2. *Exponential rate of convergence of optimal solutions of the sampled problems*

In the previous subsection, we showed that we can expect to get an optimal solution for the original problem (1) by solving the sample average approximation (2) if we choose a large sample size. In this section we show that the probability of getting an optimal solution this way approaches 1 exponentially fast as we increase the sample size. We use

large deviations theory and a result due to Dai, Chen, and Birge (2000) to prove our statement. Vogel (1988) shows, under weaker conditions, that the feasible region of a sample average approximation for a chance constraint problem approaches the true feasible region at a polynomial rate and conjectures, without giving a proof, that an exponential rate of convergence is attainable under similar conditions to those we impose.

The following theorem is an intermediate result from theorem 3.1 in Dai, Chen, and Birge (2000).

Theorem 6. Let $H:\mathbb{R}^p \times \mathbb{Z} \to \mathbb{R}$ and assume that there exist $\gamma > 0$, $\theta_0 > 0$ and $\eta:\mathbb{Z} \to \mathbb{R}$ such that

$$|H(y,\xi)| \leqslant \gamma\eta(\xi), \qquad E[e^{\theta\eta(\xi)}] < \infty,$$

for all $y \in \mathbb{R}^p$ and for all $0 \leqslant \theta \leqslant \theta_0$, where ξ is an integer valued random variable. Then for any $\delta > 0$, there are $a > 0, b > 0$, such that for any $y \in \mathbb{R}^p$

$$P[|h(y) - \bar{h}_n(y)| \geqslant \delta] \leqslant ae^{-bn},$$

for all $n > 0$, where $h(y) = E[H(y,\xi)]$, and $\bar{h}_n(y)$ is a sample mean of n independent and identically distributed realizations of $H(y,\xi)$.

In our setting take $H(y,\xi) = G^i(y,\xi)$ and note that $|G^i(y,\xi)| \leqslant N^i(\xi)$, where N^i is the number of calls received in period i. If the arrival process is, for example, a (non-homogeneous or homogeneous) Poisson process, which is commonly used to model incoming calls at a call center, then N^i satisfies the condition of theorem 6 since it is a Poisson random variable, which has a finite moment generating function.

Before we prove the exponential rate we prove a lemma that shows that for any n, $Y_0^* \subseteq Y_n^* \subseteq Y^*$, precisely when all the solutions in Y_0^* are feasible for the sample average approximation, and all infeasible solutions for (6) that are equally good or better, i.e., are in the set Y_1, are also infeasible for (7).

Lemma 7. Let $n > 0$ be an arbitrary integer. The properties

1. $\bar{g}_n(y) \geqslant 0 \; \forall y \in Y_0^*$, and
2. $\bar{g}_n(y) \not\geqslant 0 \; \forall y \in Y_1$

hold if and only if $Y_0^* \subseteq Y_n^* \subseteq Y^*$.

Proof. Suppose properties 1 and 2 hold. Then by property 1 all $y \in Y_0^*$ are feasible for (7) and the optimal value of (7) is at most F^*. By property 2 there are no solutions with a lower objective that are feasible for (7), so $Y_0^* \subseteq Y_n^*$. By property 2, no solutions outside Y^* with objective value equal to F^* are feasible for (7). Hence, $Y_0^* \subseteq Y_n^* \subseteq Y^*$.

Suppose $Y_0^* \subseteq Y_n^* \subseteq Y^*$. Then F^* is the optimal value for (7). Now, since all $y \in Y_0^*$ are optimal for (7) they are also feasible for (7) and property 1 holds. All $y \in Y_1$ are infeasible for (7) since $Y_n^* \subseteq Y^*$ and therefore property 2 holds. □

Theorem 8. Suppose $G^i(y, \xi)$ satisfies the assumptions of theorem 6 for all $i \in \{1, \ldots, p\}$ and that Y_0^* is nonempty. Then there exist $\alpha > 0$, $\beta > 0$ such that

$$P\left[Y_0^* \subseteq Y_n^* \subseteq Y^*\right] \geq 1 - \alpha e^{-\beta n}.$$

Proof. Define

$$\delta_1 := \min_{y \in Y_0^*} \min_{i \in \{1, \ldots, p\}} \left\{g^i(y)\right\},$$

$$i(y) := \arg \max_{i \in \{1, \ldots, p\}} \left\{-g^i(y)\right\},$$

$$\delta_2 := \min_{y \in Y_1} \left\{-g^{i(y)}(y)\right\}, \quad \text{and}$$

$$\delta := \min\{\delta_1, \delta_2\}.$$

Here $\delta_1 > 0$ is the minimal amount of slack in the constraints "$g(y) \geq 0$" for any solution $y \in Y_0^*$. Similarly $\delta_2 > 0$ is the minimal violation in the constraints "$g(y) \geq 0$" induced by any solution $y \in Y_1$. Thus,

$$\begin{aligned}
&P\left[Y_0^* \subseteq Y_n^* \subseteq Y^*\right] \\
&\quad = P\left[\bar{g}_n(y) \geq 0 \ \forall y \in Y_0^*,\ \bar{g}_n(y) \not\geq 0 \ \forall y \in Y_1\right] && (8) \\
&\quad = 1 - P\left[\bar{g}_n(y) \not\geq 0 \text{ for some } y \in Y_0^* \text{ or } \bar{g}_n(y) \geq 0 \text{ for some } y \in Y_1\right] \\
&\quad \geq 1 - \sum_{y \in Y_0^*} \sum_{i=1}^{p} P\left[\bar{g}_n^i(y) < 0\right] - \sum_{y \in Y_1} P\left[\bar{g}_n(y) \geq 0\right] && (9) \\
&\quad \geq 1 - \sum_{y \in Y_0^*} \sum_{i=1}^{p} P\left[\left|\bar{g}_n^i(y) - g^i(y)\right| \geq \delta\right] \\
&\qquad - \sum_{y \in Y_1} P\left[\left|\bar{g}_n^{i(y)}(y) - g^{i(y)}(y)\right| \geq \delta\right] && (10) \\
&\quad \geq 1 - \sum_{y \in Y_0^*} \sum_{i=1}^{p} a_i e^{-b_i n} - \sum_{y \in Y_1} a_{i(y)} e^{-b_{i(y)} n} && (11) \\
&\quad \geq 1 - \alpha e^{-\beta n}.
\end{aligned}$$

Here

$$\alpha = |Y_0^*| \sum_{i=1}^{p} a_i + \sum_{y \in Y_1} a_{i(y)} \quad \text{and} \quad \beta = \min_{i \in \{1, \ldots, p\}} b_i,$$

where $|Y_0^*|$ is the cardinality of the set Y_0^*. Equation (8) follows by lemma 7. Equation (9) is Boole's inequality. Equation (10) follows since $P[\bar{g}_n(y) \geq 0] \leq P[\bar{g}_n^{i(y)}(y) \geq 0]$ and $g^i(y) \geq \delta_1 \geq \delta$ for $y \in Y_0^*$ and $g^{i(y)}(y) \geq \delta_2 \geq \delta$ for $y \in Y_1$. Equation (11) follows from theorem 6. □

The case where Y_0^* is empty but Y^* is not would almost certainly never arise in practice. But in such a case one can solve an ε-perturbation of (2) as described at the end of section 4.1, and the results of theorem 8 hold for $0 < \varepsilon < \delta$.

5. Numerically checking the concavity of a function

The success of the cutting plane algorithm relies on concavity of each component of the service level function $\bar{g}_n$. If a component of $\bar{g}_n$ is not concave, then the algorithm may "cut off" a portion of the feasible set and terminate with a nonoptimal solution. In each iteration of the algorithm we obtain new information about $\bar{g}_n$. To improve the robustness of the algorithm, we would like to ensure that the information we receive is consistent with the notion that each component of $\bar{g}_n$ is concave.

There are 2 cases to consider. The first is where the vectors $\bar{q}_n^i(y)$ as returned by the simulation are guaranteed to be subgradients of $\bar{g}_n^i$ *if* $\bar{g}_n^i$ is concave. For example, this would occur if the vectors were exact gradients of the function $\bar{g}_n^i$ at y (assuming that it had a differentiable extension to $\mathbb{R}^p$ from $\mathbb{Z}^p$). In this case there is an easy test for nonconcavity, as we will see. The second case, that appears more likely to occur in practice, is where the vectors $\bar{q}_n^i(y)$ are obtained using some heuristic, and are therefore not guaranteed to be subgradients, even if $\bar{g}_n^i$ is indeed concave. In this case, we may decide to disregard some of the potentially-unreliable "subgradient" information and focus only on the function values themselves. (This setting may also be useful if one does not have "subgradient" information at all points, as arises using the finite-differencing heuristic mentioned earlier. When evaluating the "subgradient" at y, we also compute the function value, *but not gradient information*, at points of the form $y + e_i$ where e_i is the usual ith basis vector.) If the function values themselves are consistent with the notion that the function is concave, then we may view our heuristically-derived "subgradients" with some suspicion, and even drop some of them from the optimization. An alternative would be to attempt to restrict the feasible region to a region where the functions are concave. We view the analysis of the cutting plane algorithm under these conditions as beyond the scope of this paper, partly because it is then possible that we then need to deal with the usual difficulties of nonconvex optimization. If the function values alone suggest nonconcavity, then the algorithm results should be viewed with some caution. Indeed, values reported as optimal by the algorithm could, in this case, be nonoptimal. The ability to detect when the key assumption of the cutting plane algorithm may not apply is, we believe, a strength of our approach.

Of course, one may either implement a check for nonconcavity either inline on each iteration of the cutting plane algorithm, or after the algorithm halts, or not at all. The choice depends on how conservative one wishes to be, and is therefore application dependent, and so we do not enter into a discussion of which approach to take here.

To simplify the presentation, let us consider the concavity of a real-valued function $f : \mathbb{R}^p \to \mathbb{R}$ instead of $\bar{g}_n^i$. Hopefully no confusion will arise since the previously-defined function f plays no role in this section. We assume that we are given a set of points $y^1, y^2, \ldots, y^k \in \mathbb{R}^p$ and their corresponding function values

$f(y^1), f(y^2), \ldots, f(y^k)$. The tests below allow one to conclude that either f is non-concave, or that there exists a concave function that matches the given function values. Of course, the tests cannot conclude that f is concave unless they examine all points in its domain, so that the conclusions that these tests reach are the best possible in that sense.

5.1. *Concavity check with function values and "subgradients"*

Suppose that we know the vectors $q(y^1), q(y^2), \ldots, q(y^k)$ in addition to the set of points and their function values. Here $q(y^v)$ should have the property that *if* f is concave, then $q(y^v)$ is a subgradient at y^v $(v = 1, \ldots, k)$. If they are in fact subgradients then they need to satisfy (3), i.e., all k points must lie below the k hyperplanes defined by the $q(y^v)$'s and the corresponding function values. This means that for each point y^v, $v \in \{1, \ldots, k\}$, we must check that

$$f(y^w) \leqslant f(y^v) + q(y^v)^T(y^w - y^v) \quad \forall w \in \{1, \ldots, k\}. \tag{12}$$

If this inequality is violated by some v and w, then we conclude that f is not concave in y. Otherwise, the known values of f do not contradict the concavity assumption and

$$h(y) := \inf_{v \in \{1, \ldots, k\}} f(y) + q(y^v)^T(y - y^v)$$

is a concave function (see theorem 5.5 in Rockafellar (1970)), such that $h(y^w) = f(y^w)$ $\forall w \in \{1, \ldots, k\}$. In other words if (12) holds $\forall v \in \{1, \ldots, k\}$ then a concave function exists that agrees with the observed function values and "subgradients" $q(y^v)$, $v = 1, \ldots, k$.

When this test is implemented in the framework of algorithm 1, where in each iteration k we obtain y^{k+1}, $\bar{g}_n^i(y^{k+1})$ and $\bar{q}_n^i(y^{k+1})$, we need only check that (for each period i) the new point lies below all the previously defined hyperplanes and that all previous points lie below the hyperplane defined by the new "subgradient."

5.2. *Concavity check with function values only*

Now consider the case when only f is known at a finite number of points.

We want to know whether or not there is a concave function, say h, which passes through f at all the given points. If such a function does not exist then we conclude that f is not concave. (This problem appeared in Murty (1988, p. 539).)

We present a method where we solve a linear program (LP) and draw our conclusions based on the results of the LP. The idea behind this method is that if a one-dimensional function is concave then it is possible to set a ruler above each point and rotate it until the function lies completely below the ruler. This can also be done when dealing with functions of higher dimensions, and then the ruler takes the form of a plane $(p = 2)$ or a hyperplane $(p > 2)$.

The LP changes the given function values so that a supporting hyperplane for the convex hull of the points can be fitted through each point. The objective of this LP is to minimize the change in the function values that needs to be made to accomplish this goal.

If the changes are measured in the L_1- or L_∞-norm then the objective function is linear. The LP also gives an idea of how far, in some sense, the function is from being concave if a concave function cannot be fitted through the given points. If a concave function can be fitted then the LP will return such a function, namely the pointwise minimum of the hyperplanes computed by the LP.

It is most straightforward to use the L_1-norm to measure the changes in the function values. Then the LP can be formulated as follows:

$$\begin{aligned} &\min \sum_{v=1}^{k} |b_v| \\ &\text{subject to} \\ &\quad a_{0v} + \left(a^v\right)^T y^v = f\left(y^v\right) + b_v \quad \forall v \in \{1, \ldots, k\}, \\ &\quad a_{0v} + \left(a^v\right)^T y^w \geqslant f\left(y^w\right) + b_w \quad \forall v \in \{1, \ldots, k\}\ \forall w \in \{1, \ldots, k\},\ w \neq v. \end{aligned} \tag{13}$$

To linearize the objective function we adopt the standard trick of writing $b_v = b_v^+ - b_v^-$ and replace $|b_v|$ with $b_v^+ + b_v^-$, where b_v^+ and b_v^- are nonnegative. The decision variables are:

$a_{0v} \in \mathbb{R}$, $v \in \{1, \ldots, k\}$: intercepts of the hyperplanes,
$a^v \in \mathbb{R}^p$, $v \in \{1, \ldots, k\}$: slopes of the hyperplanes and
$b_v^+, b_v^- \in \mathbb{R}$, $v \in \{1, \ldots, k\}$: change in the function values.

The number of variables in this LP is $k(p + 1) + 2k = k(p + 3)$ and the number of constraints is $k + k(k - 1) = k^2$. We could split the LP up into k separate linear programs if that would speed up the computations, as might occur if we could run them on multiple processors in parallel, or if the LP solver was unable to detect the separable structure in this problem and exploit it. Here, the vth separate linear program tries to fit a hyperplane through the point $(y^v, f(y^v))$ that lies above all other points.

The LP is always feasible, since a feasible solution is given by $a^v = 0$, $a_{0v} = 0$ and $b_v = -f(y^v)$ for all $v \in \{1, \ldots, k\}$. It is also bounded below by 0, since the objective function is a sum of absolute values. Therefore, this problem has a finite minimum. If the minimum value is 0, then the function defined by

$$h(y) := \inf_{v=1,\ldots,k} a_{0v} + \left(a^v\right)^T y$$

is concave and $f(y^v) = h(y^v)$ for all $v \in \{1, \ldots, k\}$. On the other hand, if f is indeed concave, then there exists a subgradient at every point of f (see theorem 3.2.5 in Bazaraa, Sherali, and Shetty (1993)) and hence the constraints of the LP can be satisfied with $b_v = 0$ for all $v \in \{1, \ldots, k\}$. We have proved the following result.

Theorem 9. Consider the LP (13).

1. If the optimal objective value of the LP is 0 then there exists a concave function $h(y)$ such that $h(y^v) = f(y^v)$ for all $v \in \{1, \ldots, k\}$.
2. If f is concave then the optimal objective value of the LP is 0.

So we see that a necessary condition for f to be concave is that the optimal objective value of the LP (13) is zero. Thus we have the following corollary.

Corollary 10. If the optimal objective value of the LP (13) is positive, then f is not concave.

Note that the hyperplanes obtained from the LP are generally not subgradients of f, so we cannot use them in algorithm 1 as such. Hence, we have to solve this LP after step 2 in each iteration, or as a check after the algorithm terminates. Given the computational demands of the cutting plane algorithm, repeatedly solving this LP in each iteration does not represent a significant increase in computational effort.

6. Computational study

In this section we present a small numerical example that showcases the properties of our method. The example is far from being a realistic representation of a call center, but captures many issues in setting call center staffing levels. We will study 3 aspects of the problem in the context of the example:

1. Convergence of the cutting plane algorithm and the quality of the resulting solution.
2. Dependence of the service level in one period on staffing levels in other periods. This is of particular practical interest since traditional methods assume independence between periods.
3. Concavity of $\bar{g}_n(y)$.

Our implementation creates the integer programs (5) in AMPL and uses the CPLEX solver to solve them in step 1 of the algorithm, and a simulation model built in ProModel to perform steps 2 and 3. We used Microsoft Excel to pass data between the simulation and optimization components and to run the iterations of the algorithm. The implementation was exactly as described in algorithm 1 except for the initialization, where we started with y^1 at the lower bounds described in section 3 instead of starting with D_1 and d_1 empty.

6.1. Example

We consider an $M(t)/M/s(t)$ queue with $p = 5$ periods of equal length of 30 minutes. We let the service rate be $\mu = 4$ customers/hour. The arrival process is a nonhomogeneous Poisson process with the arrival rate a function of the time t in minutes equal to $\lambda(t) = \lambda(1 - |t/150 - 0.65|)$, i.e., the arrival rate is relatively low at the beginning of the first period, then increases linearly at rate λ until it peaks partway through the fourth period and decreases at rate λ after that. We set $\lambda = 120$ customers/hour, which makes the average arrival rate over the 5 periods equal 87.3 customers/hour.

Table 1
The iterates of the algorithm and the resulting service level function values and their 95% confidence intervals (CI). $f(y^k)$ is the objective value at y^k.

k	y^k					$\bar{g}_{100}(y^k) \pm 95\%$ CI half width (% of calls received that are answered in less than 90 sec.)					$f(y^k)$
1	11	19	27	30	29	0.4 ± 1.0 (81.5%)	-1.1 ± 2.2 (77.2%)	-1.8 ± 3.1 (76.6%)	-3.5 ± 3.2 (73.8%)	-2.1 ± 2.3 (75.5%)	125.0
2	11	21	27	33	29	0.5 ± 0.9 (81.9%)	3.4 ± 1.5 (88.5%)	0.2 ± 2.6 (80.5%)	4.1 ± 2.2 (87.4%)	-0.3 ± 2.7 (79.4%)	127.5
3	11	21	27	34	29	0.5 ± 0.9 (81.9%)	3.4 ± 1.5 (88.5%)	0.4 ± 2.6 (80.7%)	5.8 ± 1.8 (90.4%)	0.0 ± 2.6 (80.0%)	128.0

The goal is to answer 80% of received calls in each period in less than 90 seconds. The customers form a single queue and are served on a first come first serve basis. If a server is still in service at the end of a period it finishes that service before becoming unavailable. For example, if there are 8 busy servers at the end of period 3 and period 4 only has 6 servers then the 8 servers will continue to serve the customers already in service, but the next customer in the queue will not get service until 3 customers have finished service.

There are 6 permissible tours, including 4 tours that cover 2 adjacent periods, i.e., periods 1 and 2, 2 and 3, 3 and 4, and finally 4 and 5. The remaining 2 tours cover only one period, namely the first and the last. The cost of the tours covering 2 periods is \$2 and the single period tours cost \$1.50.

6.2. *Results*

We selected a sample size of $n = 100$ for running the algorithm. The lower bounds on y are depicted in the row $k = 1$ in table 1. Note that the staffing levels at the lower bounds result in an unacceptable level of service and thus a method which would treat the periods independently, would give an infeasible solution, since the service level is as low as 73.8% in period 4. The algorithm terminates after only 3 iterations with an optimal solution to the sample average approximation. To verify that this is indeed an optimal solution we ran a simulation for all staffing levels that have lower costs than the optimal solution and satisfy the initial lower bounds. None of these staffing levels satisfied $\bar{g}_{100}(y) \geq 0$, so the solution returned by the algorithm is the optimal solution for the sample average approximation. By including the 95% confidence interval we get information about the quality of the solution as a solution of the original problem. In the example, the confidence intervals in periods 1, 3 and 5 cover zero, which is a concern since we cannot say with conviction that our service level is acceptable in those periods. To get a better idea of whether the current solution is feasible for the original problem we calculated $\bar{g}_{999}(y^3) = (0.5 \pm 0.3, 3.0 \pm 0.5, 2.3 \pm 0.7, 5.1 \pm 0.7, 0.0 \pm 0.8)^T$, so we are more confident that the service levels in periods 1 and 3 are acceptable. The service level in period 5 is close to being on the boundary, hence our difficulty in determining

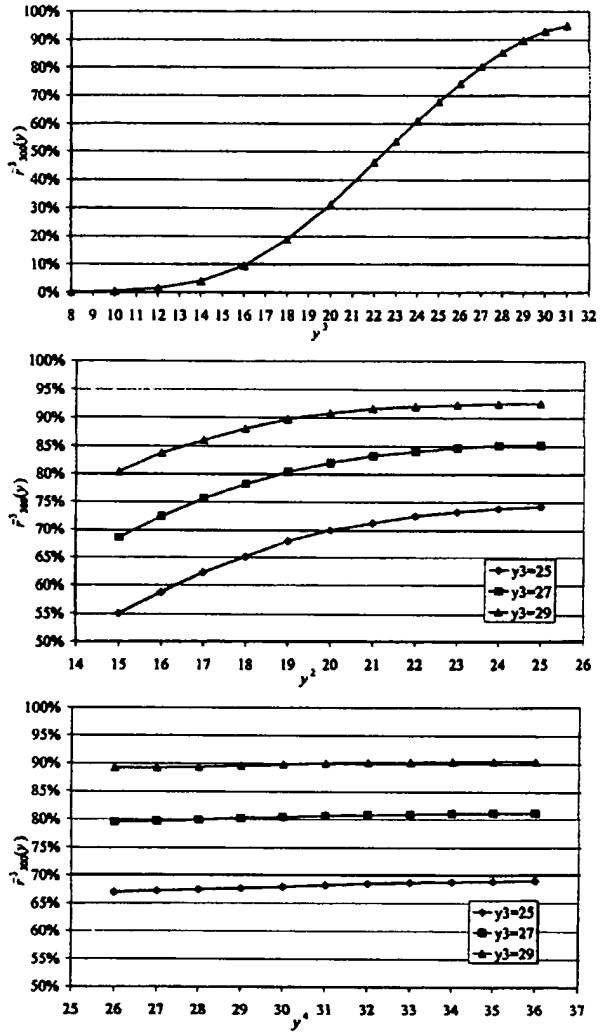


Figure 1. Dependence of staffing levels on the service level in period 3 of the example in section 6.1.

whether the solution is feasible or not. From a practical standpoint, if we are infeasible, then we are so close to being feasible that it probably is of little consequence.

We already noted that there is dependence between periods. To investigate the dependence further we calculated $\bar{r}_n^3(y)$, the percentage of calls received in period 3 answered in less than 90 seconds, i.e., $\bar{r}_n^3(y) := \sum_{d=1}^n S^3(y, \xi^d) / \sum_{d=1}^n N^3(y, \xi^d)$. We chose period 3 to demonstrate how the service level depends on staffing level in both the period before and after. Figure 1 illustrates this point. The graphs show the service level

Table 2
Concavity study. Low, medium and high staffing levels in each period and the optimal values of (13).

Period	1	2	3	4	5
Low	10	18	26	29	29
Medium	12	20	28	31	31
High	14	22	30	33	33
Optimal value	0.0	0.0	$4.0 \cdot 10^{-3}$	$1.5 \cdot 10^{-2}$	$5.7 \cdot 10^{-4}$

in period 3 as a function of the staffing level in period 3 (1.a), period 2 (1.b) and period 4 (1.c) when the staffing levels in other periods are fixed. The service level depends more on the staffing level in the period before than the period after as could be expected. That is because a low staffing level in an earlier period results in a queue buildup, which increases waiting in the next period. The reason why the staffing level in a later period affects the service level in an earlier period is that customers that called in the earlier period may still be waiting at the beginning of the next period and thus receive service earlier if there are more servers in that period. We noted dependence between periods as far apart as from the first period to the last. Figure 1 also supports the concavity assumption of the service level function when y is within a region of reasonable values, i.e., at least satisfies some lower bounds. It is, however, clear that the service level function looks like an s-shaped function over a broader range of y's as pointed out by Ingolfsson and Cabral (2002). That would not be problematic if one were to include the aforementioned lower bounds on y and if the concavity assumption holds for all y above the lower bounds.

We also performed a separate concavity check based on the method in section 5.2. In an effort to demonstrate these ideas as clearly as possible we performed the concavity check *outside* the scope of the cutting plane algorithm itself, using a selection of points that appear reasonable from a performance standpoint. We used a sample size of 300 and calculated $\bar{g}_{300}(y)$ at 3 different staffing levels (labelled low, medium and high in table 2) for each period, i.e., at $3^5 = 243$ points. We solved the linear program (13) for each $\bar{g}^i_{300}(y)$, $i \in \{1, \ldots, 5\}$, and obtained the results in table 2. We see that the service level functions in periods 1 and do not violate the concavity assumption at the observed points. The other functions violate the concavity condition. The values of the b_v's, i.e., the changes needed to satisfy the concavity assumption are all small, as can be seen by the objective value. We examined the points at which nonconcavity was detected, and noted that they occurred when a change in staffing level in a different period was made. (It is a strength of the LP-based concavity check that we were able to discover a region where the nonconcavity was exhibited.) The service level in period 3 increased, for example, more when the staffing level in period 1 was increased from 12 to 14 than when it was increased from 10 to 12 at staffing levels 22 and 30 in periods 2 and 3, respectively. The reason for this violation of the concavity assumption is not obvious.

One possible explanation is that our measure of service quality is binary for each customer, so that "rounding" may contribute to the nonconcavity. To elaborate, in the above example it is possible that unusually many customers exceed the waiting time limit of 90 seconds by very little when there are 12 servers in period 1, so that the effect of adding servers at this staffing level is more than when servers are added at a lower level. We would expect such a "rounding" effect to be averaged out in a longer simulation. In fact, we increased the sample size to 999 (the maximum number of replications in ProModel 4.2) and calculated the service level at the problematic points. We discovered that the nonconcavity vanished. Therefore, we make the following conjecture.

Conjecture 11. For $M(t)/M/s(t)$ queues of the form considered here there exists a finite $y_0 \geqslant 0$ such that the service level function g is nondecreasing and concave in y in the region $y \geqslant y_0$. Furthermore, $\bar{g}_n$ is nondecreasing and concave in y in the region $y_0 \leqslant y \leqslant y_1$ for all n large enough w.p. 1, for any fixed $y_1 \geqslant y_0$.

7. Conclusions

In this paper we have shown that combining simulation and cutting plane methods is a promising approach for solving optimization problems in which some of the constraints can only be assessed through simulation. As a motivating example we studied the problem of minimizing staffing costs in call centers when traditional methods fail, either because of the characteristics of the problem, or if a detailed model of the call center dynamics are desired. We performed a computational study, which supports the use of the cutting plane method and demonstrates how it can be implemented.

There are several interesting directions for future research. We established the theoretical foundation of the method, but an obvious drawback of the method is the large computational effort required to solve realistic problems. More research (in progress) is needed to make this a practical method. In relation to the integer programs one should investigate integer programming algorithms that can utilize the special structure of the relaxed problems solved in each iteration and consider allowing approximate solutions of the IPs, especially in early stages of the algorithm. One might consider Lagrangian relaxation techniques for solving these problems, still in the context of simulation and optimization, since we are approximating the constraints.

Other areas of interest include coming up with methods for obtaining subgradients or improving the current heuristic of using finite differences. It would add to the robustness of the method to study the properties of the solutions when some of the conditions set forth in this paper, e.g., the concavity of the service levels, are violated.

We only tested our algorithm on one simple example. It would be informative to run the algorithm on more complicated problems and include in the simulation model factors such as absenteeism and skill-based routing, not to mention implementing the algorithm in other types of service systems than call center staffing. We are currently pursuing many of these issues.

Acknowledgments

We would like to thank Ryan Anthony, Ernest Fung and Paul Rosenbaum, undergraduate students at Cornell University, for their contributions to the computational study. We would also like to thank the anonymous referees for their comments. This work was supported by National Science Foundation Grant DMI-9984717.

References

Akşin, O.Z. and P.T. Harker. (2001). "Modeling a Phone Center: Analysis of a Multichannel, Multiresource Processor Shared Loss System." *Management Science* 47(2), 324–336.

Bazaraa, M.S., H.D. Sherali, and C.M. Shetty. (1993). *Nonlinear Programming: Theory and Algorithms.* New York: Wiley.

Benders, J.F. (1962). "Partitioning Procedures for Solving Mixed-Variables Programming Problems." *Numerische Mathematik* 4, 238–252.

Billingsley, P. (1995). *Probability and Measure*, 3rd ed. New York: Wiley.

Birge, J.R. and F. Louveaux. (1997). *Introduction to Stochastic Programming.* Springer Series in Operations Research. New York: Springer.

Chen, B.P.K. and S.G. Henderson. (2001). "Two Issues in Setting Call Centre Staffing Levels." *Annals of Operations Research* 108(1), 175–192.

Chen, H. and B.W. Schmeiser. (2001). "Stochastic Root Finding via Retrospective Approximation." *IIE Transactions* 33(3), 259–275.

Dai, L., C.H. Chen, and J.R. Birge. (2000). "Convergence Properties of Two-Stage Stochastic Programming." *Journal of Optimization Theory and Applications* 106(3), 489–509.

Dantzig, G.B. (1954). "A Comment on Edie's "Traffic Delays at Toll Booths"." *Operations Research* 2(3), 339–341.

Dyer, M.E. and L.G. Proll. (1977). "On the Validity of Marginal Analysis for Allocating Servers in $M/M/c$ Queues." *Management Science* 23(9), 1019–1022.

Glasserman, P. (1991). *Gradient Estimation Via Perturbation Analysis.* Norwell, MA: Kluwer.

Green, L., P.J. Kolesar, and J. Soares. (2001). "Improving the SIPP Approach for Staffing Service Systems that Have Cyclic Demands." *Operations Research* 49(4), 549–564.

Healy, K. and L.W. Schruben. (1991). "Retrospective Simulation Response Optimization." In B.L. Nelson, W.D. Kelton, and G.M. Clark (eds.), *Proceedings of the 1991 Winter Simulation Conference*, pp. 901–906. Piscataway, NJ: IEEE.

Henderson, S.G. and A.J. Mason. (1998). "Rostering by Iterating Integer Programming and Simulation." In D.J. Medeiros, E.F. Watson, J.S. Carson, and M.S. Manivannan (eds.), *Proceedings of the 1998 Winter Simulation Conference*, pp. 677–683. Piscataway, NJ: IEEE.

Higle, J.L. and S. Sen. (1991). "Stochastic Decomposition: An Algorithm for Two-Stage Stochastic Linear Programs with Recourse." *Mathematics of Operations Research* 16, 650–669.

Infanger, G. (1994). *Planning under Uncertainty: Solving Large-Scale Stochastic Linear Programs.* Danvers, MA: Boyd and Fraser.

Ingolfsson, A. and E. Cabral. (2002). "Combining Integer Programming and the Randomization Method to Schedule Employees." Research Report No. 02-1, University of Alberta.

Ingolfsson, A., M.A. Haque, and A. Umnikov. (2002). "Accounting for Time-Varying Queueing Effects in Workforce Scheduling." *European Journal of Operational Research* 139, 585–597.

Jennings, O.B., A. Mandelbaum, W.A. Massey, and W. Whitt. (1996). "Server Staffing to Meet Time-Varying Demand." *Management Science* 42(10), 1383–1394.

Kelley, J.E., Jr. (1960). "The Cutting-Plane Method for Solving Convex Programs." *Journal of the Society for Industrial and Applied Mathematics* 8(4), 703–712.

Koole, G. and E. van der Sluis. (2003). "Optimal Shift Scheduling with a Global Service Level Constraint." *IIE Transactions* 35(11), 1049–1055.

Law, A.M. and W.D. Kelton. (2000). *Simulation Modeling and Analysis*, 3rd ed. Boston, MA: McGraw-Hill.

Mehrotra, A., K.E. Murphy, and M.A. Trick. (2000). "Optimal Shift Scheduling: A Branch-and-Price Approach." *Naval Research Logistics* 47(3), 185–200.

Morito, S., J. Koida, T. Iwama, M. Sato, and Y. Tamura. (1999). "Simulation-Based Constraint Generation with Applications to Optimization of Logistic System Design." In P.A. Farrington, H.B. Nembhard, D.T. Sturrock, and G.W. Ewans (eds.), *Proceedings of the 1999 Winter Simulation Conference*, pp. 531–536. Piscataway, NJ: IEEE.

Murty, K.G. (1988). *Linear Complementarity, Linear and Nonlinear Programming*. Berlin: Heldermann.

Robinson, S.M. (1996). "Analysis of Sample-Path Optimization." *Mathematics of Operations Research* 21(3), 513–528.

Rockafellar, R.T. (1970). *Convex Analysis*. Princeton, NJ: Princeton University Press.

Rubenstein, R.Y. and A. Shapiro. (1993). *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization by the Score Function Method*. Chichester: Wiley.

Shapiro, A. and T. Homem-de-Mello. (2000). "On the Rate of Convergence of Optimal Solutions of Monte Carlo Approximations of Stochastic Programs." *SIAM Journal on Optimization* 11(1), 70–86.

Thompson, G.M. (1997). "Labor Staffing and Scheduling Models for Controlling Service Levels." *Naval Research Logistics* 44(8), 719–740.

van Slyke, R.M. and R. Wets. (1969). "L-Shaped Linear Programs with Applications to Optimal Control and Stochastic Programming." *SIAM Journal on Applied Mathematics* 17(4), 638–663.

Vogel, S. (1988). "Stability Results for Stochastic Programming Problems." *Optimization* 19(2), 269–288.

Vogel, S. (1994). "A Stochastic Approach to Stability in Stochastic Programming." *Journal of Computational and Applied Mathematics* 56, 65–96.

MANAGEMENT SCIENCE
Vol. 54, No. 2, February 2008, pp. 310–323
ISSN 0025-1909 | EISSN 1526-5501 | 08 | 5402 | 0310

DOI 10.1287/mnsc.1070.0824

Staffing Multiskill Call Centers via Linear Programming and Simulation

Mehmet Tolga Cezik, Pierre L'Ecuyer
GERAD and Département d'Informatique et de Recherche Opérationnelle, Université de Montréal,
C.P. 6128, Succ. Centre-Ville, Montréal H3C 3J7, Canada
{mehmetc@amazon.com, lecuyer@iro.umontreal.ca}

We study an iterative cutting-plane algorithm on an integer program for minimizing the staffing costs of a multiskill call center subject to service-level requirements that are estimated by simulation. We solve a sample average version of the problem, where the service levels are expressed as functions of the staffing for a fixed sequence of random numbers driving the simulation. An optimal solution of this sample problem is also an optimal solution to the original problem when the sample size is large enough. Several difficulties are encountered when solving the sample problem, especially for large problem instances, and we propose practical heuristics to deal with these difficulties. We report numerical experiments with examples of different sizes. The largest example corresponds to a real-life call center with 65 types of calls and 89 types of agents (skill groups).

Key words: call centers; staffing; optimization by simulation; integer programming; cutting planes; skill-based routing; subgradient cuts
History: Accepted by Ger Koole, special issue editor; received November 11, 2004. This paper was with the authors 2 months for 2 revisions.

1. Introduction

We consider a telephone call center, or a more general contact center, where different types of calls arrive at random and different groups of agents answer these calls. Each type of call requires a specific skill and each agent group (also called skill group) has a given subset of these skills, so all agents in that group can handle the corresponding call types and only those. There may be preferences among these agent groups for a given type of call, either because some groups are better than others at handling it, or because we prefer to save certain agent groups for other call types.

The calls arrive according to arbitrary stochastic processes that could be nonstationary, and perhaps doubly stochastic (see, e.g., Avramidis et al. 2004). An arriving call can be served immediately if an agent with the appropriate skill is available, or may have to wait in a queue. An *abandonment* occurs whenever the waiting time of a call exceeds its (random) patience time. That call is then lost. *Skill-based routing* (SBR) strategies are used to determine which agent group handles each individual call.

The goal is to minimize the operating cost of the center under a set of constraints on the quality of service (QoS). The decisions to be made are how many agents of each skill group to have in the center as a function of time. In a *staffing problem*, the day is divided into periods (e.g., 30 minutes or one hour each), and one simply decides the number of agents of each group for each period. In a *scheduling problem*, a set of admissible work schedules is first specified, and the decision variables are the number of agents of each skill group in each work schedule. This determines the staffing indirectly, while making sure that it corresponds to a feasible set of work schedules. A yet more restrictive version of the problem is when there is a fixed set of available agents to be scheduled for the day or the week, where each agent has a specific set of skills. Then, we have a *scheduling and rostering problem*.

In this paper, the objective function is the sum of costs of all agents, where the cost of an agent depends on its set of skills. This cost is typically modeled as a constant plus an increment of between 5% to 20% for each additional skill. The QoS constraints are that the fraction of calls answered within a certain time limit, in the long run, exceeds a given threshold. This fraction is called the *service level*. In general, we may also have an upper bound on the abandonment ratio (the fraction of calls that are lost), or constraints on other types of performance measures. Such constraints can be imposed per call type, per period, and globally, with different thresholds. In this paper, we consider only service-level constraints.

If each agent group has a single skill, we have a system of separate parallel queues, one queue per call type. If there is a single-agent group with all skills, we have a single queue and (other things being equal) smaller waiting times than with separate queues.

However, agents with more skills are more costly, so there is a compromise to be made in choosing the skill groups. Wallace and Whitt (2005) have shown empirically, in a certain setting and for properly balanced systems with good routing strategies, that one or two skills per agent often gives a performance (in terms of QoS) that is almost as good as for a system where all agents have all skills. Note that pooling servers may also degrade QoS in certain settings, with poor routing strategies; see, e.g., Mandelbaum and Reimann (1998).

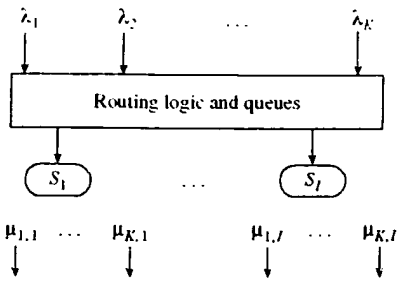
In a more general formulation, we would also want to optimize the routing of the calls. In *static routing*, each call type may have an ordered list of agent groups to try. If all are busy, the call joins a queue. There could be one queue per agent group, or one queue per call type (this is usually better because it allows more resource sharing), or one single queue for several call types, or a mixture of these. Priorities and complex routing rules can thus be implemented. For example, each agent may have an ordered list of queues to dig from when it becomes available, and the order is not necessarily the same for all agents in a group. The priorities may change based on thresholds on queue sizes or waiting times, etc. In a general *dynamic routing scheme*, the decisions may depend on the entire state of the system, which may include the current time, the number of calls of each type in service and in the queues, the elapsed service time of the calls in service, etc. Optimal dynamic schemes are usually too complicated and difficult to implement, so in practice the routing strategies are selected from a prespecified class of simpler rules. In the optimization problems studied in this paper, they are assumed to be fixed and we optimize only the staffing or scheduling. This was motivated by a specific request from a Canadian firm who wanted precisely this kind of optimization tool for the management of their call centers.

Figure 1 gives a schematic view of arrival, routing, and service in a general multiskill call center with K call types and I agent groups. The λ_ks represent the arrival rates for the different call types, the servers S_i represent the skill groups, and the mean service times $1/\mu_{k,i}$ may depend jointly on the call type and skill group.

For general background on call center management, staffing and scheduling, and multiskill centers with SBR, we refer the reader to Gans et al. (2003), Ingolfsson et al. (2003), Koole and Mandelbaum (2002), and Wallace and Whitt (2005).

Atlason et al. (2004) have proposed a general methodology, based on the cutting-plane method of Kelley Jr. (1960), to optimize the scheduling of agents in a single-call-type and single-skill call center, under service-level constraints. Their method combines simulation with integer programming and cut generation.

Figure 1 A General Multiskill Call Center



The general idea is to optimize a relaxation of a sample average version of the problem (which becomes a deterministic problem) by generating cuts from the violated service-level constraints and adding corresponding linear constraints until the optimal solution of the relaxed problem is feasible for the original problem. They solved an example with five periods and six different types of work schedules (i.e., six integer-valued decision variables).

Our aim in this paper is to extend their methodology to the multiskill setting, explore the difficulties encountered with larger problem instances, and develop (heuristic) methods to deal with these problems in a practical way. Some of the difficulties encountered are specific to the multiskill setting. With our improved methodology, we can solve (approximately) much larger problem instances than with the original methodology of Atlason et al. (2004).

The remainder of this paper is organized as follows. In §2, we formulate the staffing and scheduling problems in a multiskill center. In §3, which draws largely from Atlason et al. (2004), we first outline the solution methodology, based on simulation and cut generation. Then, we explain several difficulties encountered in applying this methodology and how we get around them to solve realistic problems. In §4, we report on numerical experiments for staffing problems over a single period, in which we assume that the system is in steady state. We solve problems of various sizes, from an artificial example with five call types and 12 skill groups, to an example of a large real-life call center with 65 call types and 89 skill groups. Section 5 contains concluding remarks. The online appendix, which is provided in the e-companion, contains additional experimental results.[1]

Staffing a call center in steady state is not quite the same as the full-scheduling problem (P1) that we formulate in the next section. However, solving large instances of it is a first step toward solving large instances of (P1). We have also solved scheduling

[1] An electronic companion to this paper is available as part of the online version that can be found at http://mansci.journal.informs.org/.

problems with several periods, with more than 100 types of work schedules, but only two types of agents, in the context of a call center operating in blend mode (with inbound and outbound calls), with the methodology described here.

2. Problem Formulation

We now give a nonlinear integer programming formulation of a scheduling problem with service-level constraints, over a time interval divided into periods. There are K call types, I skill groups, P periods, and Q types of work schedules (which we also call *shifts*). For example, the scheduling could be for one day (or week) and the periods could be successive half hours of operation of the center over that day (or week). Each shift specifies the time when the agent starts working, when he or she finishes, and all the lunch and coffee breaks.

The *cost vector* is $\mathbf{c} = (c_{1,1}, \ldots, c_{1,Q}, \ldots, c_{I,1}, \ldots, c_{I,Q})^t$, where $c_{i,q}$ is the cost of an agent of type i having shift q. The *vector of decision variables* is $\mathbf{x} = (x_{1,1}, \ldots, x_{1,Q}, \ldots, x_{I,1}, \ldots, x_{I,Q})^t$, where $x_{i,q}$ is the number of agents of type i having shift q. We use the *vector of auxiliary variables* $\mathbf{y} = (y_{1,1}, \ldots, y_{1,P}, \ldots, y_{I,1}, \ldots, y_{I,P})^t$, where $y_{i,p}$ is the number of agents of type i in period p. This vector $\mathbf{y}$ satisfies $\mathbf{y} = \mathbf{A}\mathbf{x}$, where $\mathbf{A}$ is a block diagonal matrix with I blocks $\tilde{\mathbf{A}}$, where the element (p, q) of $\tilde{\mathbf{A}}$ is 1 if shift q covers period p, and 0 otherwise.

The constraints are on the service level, defined as the fraction of calls answered within a specified time limit. This time limit was taken as 20 seconds in our computational experiments. The service level for call type k in period p is defined as

$$g_{k,p}(\mathbf{y}) = \frac{\text{E[no. of calls answered within } s_{k,p} \text{ seconds in period } p]}{\text{E[no. of calls in period } p]}$$

for some constant $s_{k,p}$. It is equal (with probability one) to the fraction of calls of type k answered within $s_{k,p}$ seconds over an infinite number of independent and identically distributed (i.i.d.) copies of period p. This fraction generally depends on the staffing over all periods, up to period p, because reducing the staffing over a given period can increase the queue lengths and thus reduce the service level in the periods that follow, and may also increase the waiting time of calls answered in this given period but that arrived in earlier periods. This is why $g_{k,p}$ is written as a function of $\mathbf{y}$. In our model, the abandonments that occur before the time limit $s_{k,p}$ are not counted, whereas the calls that abandon after the time limit are counted in the total. The service-level constraints are of the form $g_{k,p}(\mathbf{y}) \geq l_{k,p}$, where each constant $l_{k,p}$ belongs to the interval $[0, 1)$.

The model also has constraints on the aggregate service level over a given call type, a given period, and overall. For example, the aggregate service level over call type k is the expected total number of calls of type k answered within some time limit s_k over the day (say), divided by the expected total number of calls of type k received over the day. We denote by $g_p(\mathbf{y})$, $g_k(\mathbf{y})$, and $g(\mathbf{y})$ the aggregated service levels for period p, call type k, and overall, respectively. The corresponding time limits are s_p, s_k, and s, and the corresponding minimal service levels are l_p, l_k, and l.

These functions $g_\bullet$ (the ratios of expectations) are generally unknown and much too complicated to be evaluated exactly. They can be either *approximated* via simplified queueing models (e.g., as in Ingolfsson et al. 2003) or *estimated* by simulation. In this paper, we concentrate on the second alternative.

With this notation, we define the *scheduling problem* as

$$\begin{aligned}
\min\ & \mathbf{c}^t\mathbf{x} = \sum_{i=1}^{I}\sum_{q=1}^{Q} c_{i,q}x_{i,q} \\
\text{subject to}\ & \mathbf{A}\mathbf{x} = \mathbf{y}, \\
& g_{k,p}(\mathbf{y}) \geq l_{k,p} \quad \text{for all } k, p, \\
& g_p(\mathbf{y}) \geq l_p \quad \text{for all } p, \\
& g_k(\mathbf{y}) \geq l_k \quad \text{for all } k, \\
& g(\mathbf{y}) \geq l, \\
& \mathbf{x} \geq 0, \text{ and integer.}
\end{aligned} \tag{P1}$$

The *staffing problem* is a relaxation of the scheduling problem where we forget about the admissibility of schedules and just assume that any staffing is admissible. The cost vector in this setting is $\mathbf{c} = (c_{1,1}, \ldots, c_{1,P}, \ldots, c_{I,1}, \ldots, c_{I,P})^t$, where $c_{i,p}$ is the cost of an agent of group i in period p. We have

$$\begin{aligned}
\min\ & \mathbf{c}^t\mathbf{y} = \sum_{i=1}^{I}\sum_{p=1}^{P} c_{i,p}y_{i,p} \\
\text{subject to}\ & g_{k,p}(\mathbf{y}) \geq l_{k,p} \quad \text{for all } k, p, \\
& g_p(\mathbf{y}) \geq l_p \quad \text{for all } p, \\
& g_k(\mathbf{y}) \geq l_k \quad \text{for all } k, \\
& g(\mathbf{y}) \geq l, \\
& \mathbf{y} \geq 0, \text{ and integer.}
\end{aligned} \tag{P2}$$

In the special case where we consider *one period at a time*, we have $\mathbf{c} = (c_1, \ldots, c_I)^t$, where c_i is the cost of an agent of type i, and $\mathbf{y} = (y_1, \ldots, y_I)^t$, where y_i is the number of agents of type i. In this context, we often assume that the system is in steady state over the given period (but we may also assume

arbitrary initial conditions). The optimization problem then reduces to

$$\begin{aligned} \min \quad & \mathbf{c}^t\mathbf{y} = \sum_{i=1}^{I} c_i y_i \\ \text{subject to} \quad & g_k(\mathbf{y}) \geq l_k \quad \text{for all } k, \\ & g(\mathbf{y}) \geq l, \\ & \mathbf{y} \geq 0, \text{ and integer.} \end{aligned} \tag{P3}$$

Our numerical illustrations in §4 are with problem (P3) for a steady-state system. Solving this problem in itself can bring useful insight into the effect of skill-set selection and routing strategies on the service level.

It is frequent practice in call centers to solve the single-period staffing problem independently for each period, under a steady-state assumption (first step), and then find an admissible schedule that covers the required staffing at minimal cost (second step). When this approach is used, we usually have $l_p = l_k = l_{p,k}$ for all p and k, so the constraints on the aggregated levels are automatically satisfied if the constraints of (P3) are satisfied for each period and if the periods are assumed independent. Often, in practice, the constraints are only on the aggregated service level over all the periods, and we use these same constraints in all copies of (P3), i.e., for each period. This two-step approach is definitely suboptimal and the suboptimality gap can be significant (e.g., Jennings et al. 1996), but it is nevertheless often used because the original scheduling problem (P1) is deemed too hard to solve. If the staffing vector $\mathbf{y}^*$ represents the optimal solution obtained in the first step, and if we assume that this $\mathbf{y}^*$ automatically satisfies the constraints of the original problem (i.e., that the time-aggregated constraints are redundant), then the problem to solve in the second step is

$$\begin{aligned} \min \quad & \mathbf{c}^t\mathbf{x} = \sum_{i=1}^{I}\sum_{q=1}^{Q} c_{i,q} x_{i,q} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \geq \mathbf{y}^*, \\ & \mathbf{x} \geq 0, \text{ and integer.} \end{aligned} \tag{P4}$$

To solve any of these problems, we need to approximate or estimate the functions $g_\bullet$. For this, we tried variations of a queueing approximation proposed by Koole and Talim (2000) and Koole et al. (2003), where the model is approximated by a loss system (with several simplifications) and a heuristic is used to infer the service levels in the original model from the loss ratios in the loss model. We compared the results with those of a detailed simulation model and found significant differences in the service levels (more than 100% in some examples). In the remainder of this paper, we use only simulation to estimate the service level.

3. General Methodology

In this section, we summarize the main ideas of the general methodology proposed by Atlason et al. (2004), adapt it to the multiskill setting, point out several difficulties encountered with this methodology (many of them were already recognized in Atlason et al. 2004), and propose practical heuristics that can make the methodology viable for realistic problem instances.

3.1. Optimization of a Sample Problem

The service levels involved in the constraints are estimated by simulation. Suppose that we simulate the center n times, independently, over its P periods of operation. Let ω represent the source of *randomness*, i.e., the sequence of all independent $U(0,1)$ random variates that drive the successive simulation runs (regardless of their number). We may assume that ω is divided into long disjoint subsequences so that the ith simulation run always starts using random numbers at the beginning of the ith subsequence regardless of the model parameters and staffing vector. This is implemented by using random number packages that provide multiple streams and substreams (L'Ecuyer et al. 2002, L'Ecuyer 2004).

The *empirical service level* over n simulation runs, defined as the observed number of calls answered within the time limit divided by the total number of calls, is a function of the staffing level $\mathbf{y}$ and of ω. We denote it by $G_{n,k,p}(\mathbf{y}, \omega)$ for call type k in period p; $G_{n,p}(\mathbf{y}, \omega)$ aggregated over period p; $G_{n,k}(\mathbf{y}, \omega)$ aggregated for call type k; and $G_n(\mathbf{y}, \omega)$ aggregated overall. For a fixed ω, these are all deterministic functions of $\mathbf{y}$. To compute these functions at different values of $\mathbf{y}$, we simply use simulation with common random numbers, i.e., make sure that the same random numbers are used at the same place for all values of $\mathbf{y}$ (Law and Kelton 2000).

If we replace the functions $g_\bullet$ in the optimization problems by their sample counterparts $G_\bullet$, we obtain the following *empirical scheduling problem*, which is a sample average version of the original problem:

$$\begin{aligned} \min \quad & \mathbf{c}^t\mathbf{x} = \sum_{i=1}^{I}\sum_{q=1}^{Q} c_{i,q} x_{i,q} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{y}, \\ & G_{n,k,p}(\mathbf{y}) \geq l_{k,p} \quad \text{for all } k, p, \\ & G_{n,p}(\mathbf{y}) \geq l_p \quad \text{for all } p, \\ & G_{n,k}(\mathbf{y}) \geq l_k \quad \text{for all } k, \\ & G_n(\mathbf{y}) \geq l, \\ & \mathbf{x} \geq 0, \text{ and integer.} \end{aligned} \tag{P5}$$

A similar formulation can be given for the other problems, replacing g by G everywhere. The single-period staffing problem becomes

$$\begin{aligned} \min \quad & \mathbf{c}^t\mathbf{y} = \sum_{i=1}^{I} c_i y_i \\ \text{subject to} \quad & G_{n,k}(\mathbf{y}) \geq l_k \quad \text{for all } k, \\ & G_n(\mathbf{y}) \geq l, \\ & \mathbf{y} \geq 0, \text{ and integer.} \end{aligned} \tag{P6}$$

We insist on the fact that when ω is fixed, these sample problems become purely deterministic. These are the problems we solve, instead of the original problems (P1) to (P3).

We can study the convergence of the optimal solution of the sample problem to that of the original problem as $n \to \infty$, under the assumption that ω is really an infinite sequence of i.i.d. random variables (as opposed to pseudorandom numbers). This was done by Atlason et al. (2004) for their model, and earlier, e.g., by Vogel (1994) in a more general setting. Our argumentation is similar but simplified.

In our context, it is reasonable to assume that the set $\mathcal{Y}$ of potential solutions $\mathbf{y}$ is finite (e.g., it suffices to impose an upper bound on the total number of agents in the center). Let $\mathcal{Y}^*$ be the set of optimal solutions of the exact problem. Another reasonable assumption is that no service-level constraint is satisfied exactly in the original problem for these solutions $\mathbf{y} \in \mathcal{Y}^*$. Let $\mathcal{Y}_n^*$ be the set of optimal solutions of the sample problem based on n simulation runs. Each random variable $G_{n,k,p}(\mathbf{y}, \omega)$ estimates the ratio of expectations $g_{k,p}(\mathbf{y})$ by the ratio of two averages of n i.i.d. random variables having the correct (positive) expectations and finite variance. Therefore, $G_{n,k,p}(\mathbf{y}, \omega)$ converges to $g_{k,p}(\mathbf{y})$ with probability one when $n \to \infty$. This means that for every $\epsilon > 0$, there is a random variable $N_0 < \infty$ such that $|G_{n,k,p}(\mathbf{y}, \omega) - g_{k,p}(\mathbf{y})| < \epsilon$ for all $n \geq N_0$. Let δ be the smallest absolute difference between an exact service level g_* and its corresponding threshold l_*, among all solutions $\mathbf{y} \in \mathcal{Y}$. In view of the above assumptions, because $\mathcal{Y}$ is finite, there is a random N_* such that $|G_{n,k,p}(\mathbf{y}, \omega) - g_{k,p}(\mathbf{y})| < \delta$ for all $n \geq N_*$, all $\mathbf{y} \in \mathcal{Y}$, all k, all p, and all aggregate service levels as well. Then, for all $n \geq N_*$, the sample problem has exactly the same set of feasible solutions (and therefore the same optimal solutions) as the exact problem. We have just proved the following:

PROPOSITION 1. *With probability one, there is an integer $N_* < \infty$ such that for all $n \geq N_*$, $\mathcal{Y}_n^* = \mathcal{Y}^*$.*

The next proposition gives a convergence result in the sense of large deviation theory, under the assumption that the service-level estimators satisfy a standard large deviation principle (see, e.g., Ellis 1985, Shwartz and Weiss 1995). This assumption holds under fairly general conditions that should practically always hold in our setting. It typically holds with $\kappa = \kappa_0\epsilon^2$ for every ϵ small enough, for some constant $\kappa_0 > 0$.

ASSUMPTION 1. *For every $\epsilon > 0$, there are positive integers n_0 and κ such that for all $n \geq n_0$ and $\mathbf{y} \in \mathcal{Y}$,*

$$P(|G_{n,k,p}(\mathbf{y}, \omega) - g_{k,p}(\mathbf{y})| > \epsilon) \leq e^{-n\kappa}$$

for all k, p, and for the aggregate service levels as well.

PROPOSITION 2. *Under Assumption 1, there is a positive constant α and an integer $n^* < \infty$ such that for all $n \geq n^*$,*

$$P[\mathcal{Y}_n^* = \mathcal{Y}^*] \geq 1 - \alpha e^{-n\kappa}.$$

Take $\epsilon = \delta$ in the assumption. If the absolute error $|G_{n,k,p}(\mathbf{y}, \omega) - g_{k,p}(\mathbf{y})|$ is less than δ simultaneously for all constraints and all $\mathbf{y} \in \mathcal{Y}$, then the sample problem has exactly the same feasible set as the exact problem and therefore $\mathcal{Y}_n^* = \mathcal{Y}^*$. But the probability that this does not happen is bounded by the sum over all constraints and all values of $\mathbf{y}$ of the probabilities that it does not happen for this particular constraint and value of $\mathbf{y}$. Each such probability is bounded by $e^{-n\kappa}$ from the assumption and there are a finite number of them, so their sum must be bounded by $\alpha e^{-n\kappa}$ for some constant α.

The proof of the proposition also implies that the sample and exact problems have exactly the same set of feasible solutions with a probability that converges to one exponentially fast when $n \to \infty$. It reuses an argument from the proof of Proposition 2 of Yakowitz et al. (2000).

For solving the staffing problem (P3) in the steady-state case, we use the sample problem (P6), where the functions $G_{n,k}$ and G_n are estimated by simulating the model for n hours of operation. These functions converge to g_k and g and obey a central-limit theorem (pointwise) when $n \to \infty$ in the same way as for the multiperiod finite-horizon model, if we assume that the system has enough ergodicity. Then, if we restrict the search to a finite subset of the space of stable solutions $\mathbf{y}$, Propositions 1 and 2 hold in this case as well.

3.2. Integer Linear Programming with Cut Generation

We solve the sample staffing and scheduling problems by linear programming with cut generation, in a manner similar to Atlason et al. (2004). The idea is to relax the nonlinear service-level constraints and add progressively linear constraints until the optimal solution satisfies all service-level constraints.

We give the following explanations on cut generation with the global service-level function g, to simplify the notation. The same reasoning applies as well

to the service level per call type and/or per period, and to their sample counterparts.

At any given step of the algorithm, let $\bar{\mathbf{y}}$ denote the current solution (optimal for the relaxed problem with the current set of constraints). If $\bar{\mathbf{y}}$ satisfies all service-level constraints, then it is an optimal feasible solution for our problem and we are done (assuming that we did not cut out the optimal solution(s) when adding linear constraints). Otherwise, take a violated constraint, say $g(\bar{\mathbf{y}}) < l$, and suppose that g is concave in $\mathbf{y}$. (A discrete function g is called *concave* if the upper boundary of the convex hull of the points $(\mathbf{y}, g(\mathbf{y}))$ is a concave function.) Let $\bar{\mathbf{q}}$ be any *subgradient* of g at $\bar{\mathbf{y}}$, i.e., a vector that satisfies

$$g(\mathbf{y}) \le g(\bar{\mathbf{y}}) + \bar{\mathbf{q}}^t(\mathbf{y} - \bar{\mathbf{y}}) \tag{1}$$

for all $\mathbf{y}$. We want $g(\mathbf{y}) \ge l$, so we must have $l \le g(\mathbf{y}) \le g(\bar{\mathbf{y}}) + \bar{\mathbf{q}}^t(\mathbf{y} - \bar{\mathbf{y}})$, i.e.,

$$\bar{\mathbf{q}}^t\mathbf{y} \ge \bar{\mathbf{q}}^t\bar{\mathbf{y}} + l - g(\bar{\mathbf{y}}), \tag{2}$$

where the term on the right is a constant that can be readily computed. Adding this linear cut inequality to the constraints removes $\bar{\mathbf{y}}$ from the current set of feasible solutions, without removing any solution that is feasible for the original problem if g is concave.

Unfortunately, g is not always concave in our setting. Typically, it is concave in the areas where $\mathbf{y}$ is large, but not where $\mathbf{y}$ is small (this was already pointed out by Atlason et al. 2004; see also the next section). But it is quite reasonable to assume at least that g is nondecreasing. In that case, a subgradient $\bar{\mathbf{q}}$ must necessarily satisfy $\bar{\mathbf{q}} \ge \mathbf{0}$ because taking $\mathbf{y} = \bar{\mathbf{y}} + \mathbf{e}_j$ in (1) gives $\bar{q}_j \ge g(\mathbf{y}) - g(\bar{\mathbf{y}}) \ge 0$ for each j, and the next proposition provides a sufficient condition ensuring that constraint (2) cannot remove feasible solutions of the original problem.

PROPOSITION 3. *Suppose that g is nondecreasing everywhere in its domain and concave in the area (slice) Δ defined by the two linear inequalities*

$$\bar{\mathbf{q}}^t\bar{\mathbf{y}} \ge \bar{\mathbf{q}}^t\mathbf{y} \ge \bar{\mathbf{q}}^t\bar{\mathbf{y}} + l - g(\bar{\mathbf{y}}). \tag{3}$$

If a vector $\bar{\mathbf{q}} \ge \mathbf{0}$ satisfies (1) for $\mathbf{y} \in \Delta$, then the cut (2) cannot remove any feasible solution of the original problem.

If $\bar{\mathbf{q}} \ge \mathbf{0}$ satisfies inequality (1) in Δ, then $g(\mathbf{y}) < l$ everywhere in the interior of Δ, so the cut cannot remove any feasible solution there. We also have $g(\mathbf{y}) < l$ everywhere in the half-space where (2) is not satisfied because g is nondecreasing in that half-space, and every point $\mathbf{y}$ in that half-space satisfies $\mathbf{y} \le \mathbf{y}_1$ for some point $\mathbf{y}_1 \in \Delta$.

The area Δ is the slice of space between the hyperplane that determines the cut (2) and a parallel hyperplane that passes through the point $\bar{\mathbf{y}}$. We call a vector $\bar{\mathbf{q}} \ge \mathbf{0}$ that satisfies (1) for $\mathbf{y} \in \Delta$ a Δ-*subgradient* at $\bar{\mathbf{y}}$.

Cuts of the form (2) can be added for all violated constraints simultaneously in a single step, or for only some of the violated constraints. The next issues are: How do we obtain a subgradient, how can we make sure that g is concave, and what do we do if it is not concave? They are all addressed by heuristics.

3.3. Difficulties and Heuristic Cures

3.3.1. Getting a Subgradient. To obtain a (tentative) Δ-subgradient $\bar{\mathbf{q}}$ at $\bar{\mathbf{y}}$, we simply use forward finite differences as follows. For $j = 1, \ldots, IP$, choose an integer $d_j \ge 0$. Compute the function g at $\bar{\mathbf{y}}$ and at $\bar{\mathbf{y}} + d_j\mathbf{e}_j$ for $j = 1, \ldots, IP$, where $\mathbf{e}_j$ is the jth unit vector, with a one in position j and zeroes elsewhere. We define $\bar{\mathbf{q}}$ as the IP-dimensional vector whose jth component is

$$\bar{q}_j = [g(\bar{\mathbf{y}} + d_j\mathbf{e}_j) - g(\bar{\mathbf{y}})]/d_j \tag{4}$$

for all j. If we are sure that g is concave, taking $d_j = 1$ for all j is certainly the most natural choice, but the more general form with $d_j \ge 1$ is sometimes convenient, e.g., when the sample function is not smooth enough and/or not concave (we will return to this). If g is nondecreasing (which we assume), then $\bar{\mathbf{q}} \ge 0$. If g is concave and $d_j = 1$, then $\bar{q}_j$ is a subgradient for g viewed as a function of y_j alone (a one-dimensional function with discrete domain) for each j. Unfortunately, even under these conditions, $\bar{\mathbf{q}}$ is not necessarily a Δ-subgradient of g because (1) may fail to hold in some diagonal direction (i.e., for some vector $\mathbf{y} - \bar{\mathbf{y}}$ having several positive components). Atlason et al. (2003) pointed out this important issue and gave examples; we provide a similar discussion for completeness.

A two-dimensional function g may satisfy $g(0,0) = g(1,0) = g(0,1) = a$ and $g(1,1) > a$ and yet be concave. For example, $g(y_1, y_2)$ can be defined by two planes that match these values, one plane for $y_1 \ge y_2$ and another plane for $y_1 \le y_2$. In this case, we would get $\bar{\mathbf{q}} = (0,0)$ and this vector would not be a subgradient. The corresponding cut may then remove solutions $\mathbf{y}$ that do not satisfy (1) and do satisfy the service-level constraints of the original problem. So, we can only view $\bar{\mathbf{q}}$ as a *heuristic guess* for a subgradient. On the other hand, in our experiments, we routinely checked (1) in some positive diagonal directions, e.g., with each component of $\mathbf{y} - \bar{\mathbf{y}}$ equal to one or two, and never found any serious problem in the cases where g was concave with respect to each of its coordinates separately.

Suppose that g satisfies the *submodularity condition*

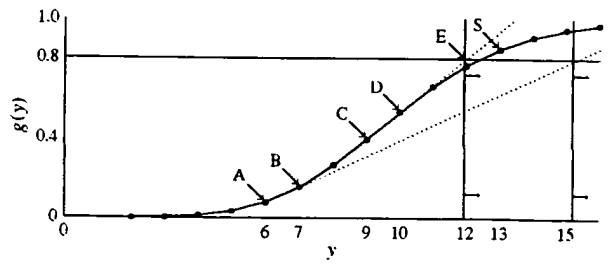
$$g(\mathbf{y} + \mathbf{d}_1 + \mathbf{d}_2) + g(\mathbf{y}) \le g(\mathbf{y} + \mathbf{d}_1) + g(\mathbf{y} + \mathbf{d}_2) \tag{5}$$

for all nonnegative integer vectors $\mathbf{d}_1$ and $\mathbf{d}_2$. For the functions g, in our setting, this condition means that adding a vector $\mathbf{d}_2$ of agents to the center brings less gain if the current (vector) number is $\mathbf{y}+\mathbf{d}_1$ than if it is only $\mathbf{y}$ (i.e., the larger the current staffing, the smaller the marginal gain of adding more agents). It is quite reasonable to expect this to be true, at least when $\mathbf{y}$ is large enough. Under condition (5), g is necessarily concave and the vector $\bar{\mathbf{q}}$ defined by (4) with $d_j = 1$ for all j always satisfies (1) for $\mathbf{y} \geq \bar{\mathbf{y}}$. However, it is not necessarily a Δ-subgradient because (1) may fail to hold for some vectors $\mathbf{y}$ such that $\bar{\mathbf{y}} - \mathbf{y}$ has both positive and negative components. This was pointed out in Atlason et al. (2003).

In general, if g is not concave or if $\bar{\mathbf{q}}$ is not a subgradient for some reason, the cut (2) may remove part of the set of feasible solutions (in terms of service level) of the original problem. If this includes the optimal solution(s), we will simply end up with a suboptimal solution. For this reason, it is generally a good idea to run the algorithm more than once with different streams of random numbers and/or slightly different parameters, and retain the best solution found.

3.3.2. Nonconcavity and Related Problems. A second (related) issue concerns the concavity of g. Typically, g is a convex function of each coordinate of $\mathbf{y}$ when these coordinates are small, and a concave function when the coordinates are large. The rationale is that if there are too few agents, almost no call is answered within the time limit and adding one agent is not likely to change this by much (at least for a large center). But if we keep adding agents, at some point the fraction $g(\mathbf{y})$ increases at a faster rate. Eventually, when there are enough agents so that most calls are answered within the time limit, adding more agents has little effect on $g(\mathbf{y})$. Figure 2 gives a one-dimensional illustration of this. The function $g(y)$ has an S-shape. A similar illustration appears in Atlason et al. (2003) and this was observed earlier by Henderson and Mason (1998) and Ingolfsson et al. (2003). For the special case of an $M/M/s$ queue with arrival rate λ and service rate μ, the concavity of the service level as a function of s was proved for $s > \lambda/\mu$ by Jagers and van Doorn (1991).

Figure 2 The S-Shaped Service-Level Function



In Figure 2, suppose that $l = 0.8$ and that we use $d_j = 1$ to get our subgradients. The optimal solution in this one-dimensional case is the smallest y for which $g(y) \geq 0.8$, namely, $y = 13$. It corresponds to point S in the figure. If $\bar{y} = 9$, where g is concave, the subgradient $\bar{q}$ is the slope of the line that passes through points C and D in the figure. This line intersects the horizontal line $g(y) = l = 0.8$ at $y = 11.95$ (point E in the figure). The corresponding cut to be added, $qy \geq q\bar{y} + l - g(9)$, can be written as $y \geq 11.95$, or $y \geq 12$ because y must be integer. This constraint removes only infeasible solutions, i.e., values of y for which $g(y) < 0.8$. On the other hand, if $\bar{y} = 6$, we see that g is not concave in that area. The subgradient $\bar{q}$ at $\bar{y} = 6$ is the slope of the line going through points A and B (the lowest dotted line), which intersects the line $g(y) = 0.8$ at $y = 15.20$. The cut-generation procedure would then add the constraint $y \geq 15.20$, or $y \geq 16$, which eliminates a large chunk of the feasible set, including the optimal solution.

Proving the joint concavity around the current solution appears much too difficult and impractical, so we only check the univariate concavity of g with respect to each coordinate y_j separately. More precisely, we check if $\bar{q}_j$ is a nondecreasing function of d_j, by verifying if

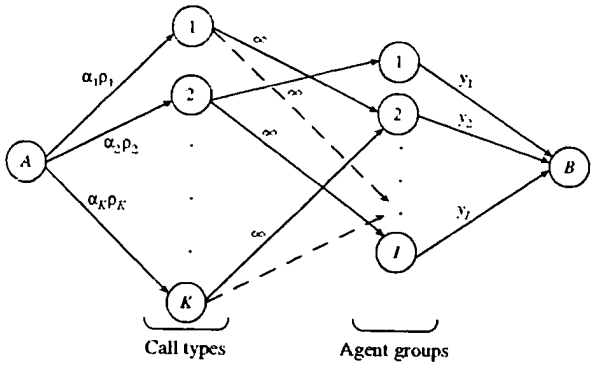
$$g(\bar{\mathbf{y}} + (d_j + 1)\mathbf{e}_j) - g(\bar{\mathbf{y}}) \leq \bar{q}_j(d_j + 1)$$

for each j. These are only necessary concavity conditions. Atlason et al. (2004) use a different type of concavity check; it is multivariate and uses all function evaluations made so far.

These (imperfect) concavity checks are expensive because they require several additional simulations (at $\bar{\mathbf{y}} + (d_j + 1)\mathbf{e}_j$). One possibility is to simply bypass them altogether, to save time, at the risk of cutting out too much of the feasibility check. To reduce that risk, it is a good idea to take larger values of d_j when the current service level is much smaller than its minimal value specified by the constraints. In our computational experiments, we found this to be more efficient than checking concavity, in the sense that it usually yields a very good solution in roughly half the time. For the experiments reported in §4, we took $d_j = 3$ when the service level corresponding to the considered cut was less than 0.5, $d_j = 2$ when it was between 0.5 and 0.65, and $d_j = 1$ when it was greater than 0.65.

3.3.3. Initial Constraints and What to Do When We Get Into Areas of Nonconcavity. At the beginning of the algorithm, if we relax all the nonlinear service-level constraints, the optimal solution of the relaxed problem is $\mathbf{y} = \mathbf{0}$. The functions g are obviously nonconcave at that point, so we are in trouble right from the start! We must find a way of eliminating areas of nonconcavity by adding other types of

Figure 3 Covering Load by Skill Supply



constraints to restrict the set of admissible solutions a priori, before generating any cut based on a subgradient. That is, impose $\mathbf{y} \in \mathcal{Y}$ for some set $\mathcal{Y}$ in which the functions $g_\bullet$ are more likely to be concave.

We obtain such constraints as follows. For any given period, we impose that the skill supply of agents during that period can cover a fraction α_k of the load of call type k during that period for all k, for some constants α_k that are usually close (or equal) to one. To explain what this means, we assume for simplicity that call type k has arrival rate λ_k and service rate μ_k (independent of the agent group) in a given period, and that the number of agents of each group for that period is given by the vector $\mathbf{y} = (y_1, \ldots, y_I)^t$. The *load* for call type k is $\rho_k = \lambda_k/\mu_k$.

Consider the graph of Figure 3, where there is an arc of infinite capacity from call type k to agent group i if and only if this agent group can handle call type k. Each arc going from an agent group i to node B on the right has a capacity equal to the number of agents in that group. Each arc going from node A to a call type k on the left has capacity equal to $\alpha_k\rho_k$, the load that we want to cover for that call type. We compute the maximal flow that can go from A to B in that graph. If we view the load of each call type as fluid, the flow that goes through call type k and agent group i corresponds to the load of call type k handled by agent group i, i.e., the number of agents of group i required to handle that load if these agents are busy 100% of their time. This is a real number (not necessarily an integer) and we call it the *skill supply* devoted to call type k. If the maximal flow equals

$$\bar{\rho} \stackrel{\text{def}}{=} \sum_{k=1}^{K} \alpha_k\rho_k,$$

this means that there is enough skill supply to handle the desired load for each call type. Otherwise, this is not possible. This is summarized in the next proposition.

PROPOSITION 4. *The maximum flow in the graph of Figure 3 is $\bar{\rho}$ if and only if there is enough skill supply in the agent groups to cover a load of $\alpha_k\rho_k$ for call type k, simultaneously for all k (for a total load of $\bar{\rho}$).*

For any given solution $\mathbf{y} = \bar{\mathbf{y}}$, the maximum flow in the graph can be computed easily by a standard max-flow algorithm. If the flow is less than $\bar{\rho}$, the algorithm finds a minimum cut that provides a valid linear inequality on $\mathbf{y}$ needed to reach the required maximum flow, but violated by the current solution. We add this constraint and find a new solution $\bar{\mathbf{y}}$. This is repeated until the maximum flow reaches $\bar{\rho}$.

This procedure is performed at the beginning of the algorithm and is repeated each time we obtain a new solution $\bar{\mathbf{y}}$ after having added linear inequalities based on service-level constraints. The computational requirement for solving this max-flow problem on a sparse bipartite network such as ours is almost insignificant in comparison with the overall computing time, regardless of the values of K and I (which may reach 100 in large call centers). In our computational experiments, we took all α_ks equal to one, except for a situation that we will describe in a moment.

It is important to underline that this procedure is a heuristic, in the sense that being able to handle the load (or a given fraction of it, as specified above) is neither necessary nor sufficient for the model to be stable. It is not necessary because abandonments can keep the system stable if the load cannot be covered. Even without abandonments, if the load is not covered during a given period, a queue will build up during that period but these calls could be handled during the next period, in which there could be extra capacity. These are the reasons why we may not always take $\alpha_k = 1$ for all k. In the case of steady-state models without abandonments, however, we should take $\alpha_k \geq 1$. But even if the maximum flow exceeds $\bar{\rho}$ with $\alpha_k > 1$ for all k, i.e., if there is enough skill supply to simultaneously cover more than the load for each call type, this does not necessarily imply that the system is stable (see Example 1 in §4, and Garnett and Mandelbaum 2000, for counterexamples). Stability in this case generally depends on the priority rules in the routing strategies. The procedure is nevertheless very useful to cut out areas of nonconcavity of the functions $g_\bullet$.

Even when the skill supply can cover the load of all call types, it may happen that certain call types have extremely low service level if the routing rules give them low priority. When we want to generate a subgradient cut based on the service-level threshold for such a call type, we run into a nonconcavity problem. The "subgradient" is often nearly flat (zero or almost zero) and the corresponding cut would remove all the good solutions. In this case (which typically happens for highly unbalanced systems, where certain

types of calls have very low priority but yet require a minimal service level), we need a different type of heuristic. What we do is simply increase the value of α_k used in the max-flow problem for the call type k having the highest service-level gap, i.e., the smallest value of $g_{k,p}(\bar{\mathbf{y}}) - l_{k,p}$ or $g_k(\bar{\mathbf{y}}) - l_k$. The trick is to increase α_k just enough to get out of the nonconcavity area. An appropriate value could be found by trial and error, checking the concavity and the service levels obtained for each trial. In our computational experiments, we adopted a heuristic that increases α_k so that the resultant service level for call type k is between 0.01 and 0.1. For the examples that we tried, this always appeared sufficient to get rid of these flat subgradients, and taking a larger α_k was counterproductive (empirically).

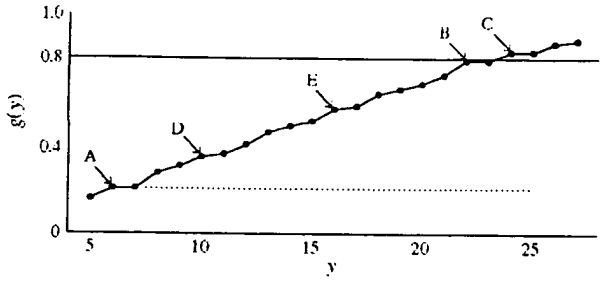
When very low service levels are obtained for certain types of calls even if the global service level is high, this may indicate poor routing strategies. In that case, significant improvements on the service level per call type can often be obtained by changing the routing rules, which is likely to be less expensive than adding more agents.

3.3.4. Cut-Generation Priorities. The problem of certain call types having zero (or near zero) service level happens more frequently when the global service level is still low as well. Moreover, when the global service level is higher, the service level for a given type of call tends to be concave over a wider range of values than when the global service level is low. We observed empirically that in many cases, when the global service level is high, the service level of a given call type is concave as a function of a given y_j practically as soon as it becomes positive.

For this reason, as long as the global service level is below a given threshold ℓ_*, we only generate cuts based on the global service-level constraint. After that threshold has been reached, at each step of the algorithm we generate and add a cut for each service-level constraint that is not satisfied. All these constraints are added simultaneously. Adding several constraints at the same time generally makes the algorithm more efficient because it reduces the total number of steps and thus the number of simulations runs. On the other hand, adding bad cuts (obtained in nonconcave areas) can be damaging. So, there is a compromise to be made and the choice of ℓ_* decides on that compromise. In our experiments, that threshold was 0.65.

3.3.5. Problems with the Smoothness of Sample Functions. Yet another difficulty comes from the fact that the sample functions $G_{n,\bullet}$ are typically less smooth than their "expected value" counterparts $g_\bullet$. This was also noticed by Atlason et al. (2004). Because of that, $G_{n,\bullet}$ is often locally nonconcave in the area where $g_\bullet$ is concave. The explanation is that for a given sample path, if we increase the number of agents one at a time, the actual number of calls that get served within the time limit may increase sometimes by a large amount, sometimes by a small amount, sometimes not at all, and the amount of increase is not necessarily monotone, for a small n, even if it becomes monotone when $n \to \infty$. When we increase the sample size n they become smoother, so the problem diminishes, but it may take a very large value of n before it disappears completely, and that value is random. Increasing n also makes the simulation much more expensive.

Figure 4 **Sample Service-Level Function of a Call Type (From the Large Example) When All Except One Agent Groups are Fixed and the Global Service Level Is Greater than 0.65**



This is illustrated in Figure 4, which shows the actual sample function $G_{n,k}$ for a given call type k as a function of the number of agents of a given group that can handle that call type, with the number of agents in all other groups held fixed, in a simulation of 50 hours of operation of the large example of §4. The sample function is flat between 6 and 7, between 22 and 23, and between 24 and 25. If we use $d_j = 1$ in this case, at $\bar{y} = 6$ and 22 (points A and B in the figure), the subgradient is zero and the corresponding cut removes everything! The subgradient is also zero at $\bar{y} = 24$ (C in the figure), but the constraint is already satisfied at that point. At D and E, the subgradient is not zero, but it is small enough so the corresponding cut would impose an excessive lower bound on y.

This type of problem can be detected by univariate concavity checks. When it occurs, we could simply use larger values of d_j in the subgradient estimators. In the example in the figure, for example, taking $d_j = 2$ or 3 would do fine. A sensible heuristic could be to try $d_j = 1$, 2, and 3, and take the one that gives the largest slope. When the service level $G_{n,\bullet}$ at the current solution is still much smaller than the threshold, it is usually a good idea to start directly with some $d_j > 1$. In our computational experiments of §4, we simply bypassed all concavity checks (to save time) and directly used larger values of d_j as explained earlier.

With all these heuristics, subgradients close to zero are still encountered once in a while and we must

avoid adding the corresponding cuts. For that, we use another threshold: when all coordinates of $\bar{\mathbf{q}}$ are smaller in absolute value than this threshold, we do not add this cut. The other cuts added at the same step will usually take care of improving the situation enough to circumvent the problem. This threshold was set at 0.01 in our experiments.

3.3.6. Relaxing Integrality and Rounding Up. For problem instances where the vector $\mathbf{y}$ has a small dimension, we can solve the sample problem at each step (i.e., after adding new constraints) by integer programming (IP). The simulation time (to evaluate the service levels required to compute $\mathbf{q}$) typically dominates the total CPU time in that case. But when the problem becomes too large, the time required for solving the integer program becomes excessive (this time increases exponentially with the size of the problem) and we must find an alternative.

What we do in that case is simply relax the integrality constraints: at each iteration, we solve the standard linear programming (LP) problem instead of the IP problem. Then, we round up each component of the solution to an integer. Obviously, all constraints satisfied by the optimal LP solution are also satisfied by the rounded-up vector. We then run simulations with this rounded-up solution to check the service-level constraints, check concavity, and estimate a subgradient.

At the last iteration, when we find a rounded-up solution that satisfies all service-level constraints, we perform a local search around that solution to see if it can be easily improved. For the numerical experiments reported here, the local search was rather simplistic: we just decreased each component of the vector $\bar{\mathbf{y}}$ by one, by decreasing order of agent cost, until we reached infeasibility. This local search procedure could certainly be improved.

We also experimented this approach on medium-sized problems, where IP was practical but yet took a significant fraction of the total CPU time. With the LP+rounding approach, the time for solving the LP was almost negligible compared with the simulation time (so the overall algorithm was faster) and the results were generally within 1% of the IP results.

4. Computational Experiments

The numerical examples considered here are for a single period only and we solve the staffing problem under the assumption that the system is in steady state. For each k, the arrivals are assumed to be from a stationary Poisson process with rate λ_k, while the service times and patience times are exponential with respective rates μ_k and ν_k. All these rates are in calls per hour.

For all experimental results reported here, the thresholds and parameters of the heuristics were selected as mentioned in the previous section. Thus, all examples were solved with the same algorithm. Our computer had a 2.0 GHz AMD Athlon 64 Processor 3,200+ running Linux, SUN Java Development Toolkit 1.4.2, and CPLEX 8.1 was our LP/IP solver. The simulations were performed using the contact center simulation package described in Buist and L'Ecuyer (2005), developed in Java in the framework of the SSJ simulation package (L'Ecuyer 2004). In our simulations, we always start the system full (all agents busy but no call in the queues). At each value of $\mathbf{y}$, we perform a single simulation run of length $T+T/20$ (in the simulation time frame) for some constant T, divide it into 21 batches of length $T/20$, and discard the first batch. The 20 other batches are used to compute estimates and confidence intervals for the service levels, globally and for each call type. Unless stated otherwise, $T = 50$ hours, so we have 2.5 hours of warm-up and 50 hours of simulation. For the smaller examples, we tried $T = 500$ hours for comparison. The results differed by less than 1% except for a single case where the longer simulation found a much better solution (see Example 1). The total computing times are approximately proportional to T.

4.1. Staffing a Center of Moderate Size

We first consider two examples of moderate size, one with five call types and 12 agent groups, the other with 20 call types and 15 agent groups. Example 1 is adapted from Koole and Talim (2000).

For the routing, we assume that each call type has an ordered list of agent groups, used to select an available agent upon arrival. If all agent groups in that list are busy, the call joins a queue (one FIFO queue per call type). Each agent group has an ordered list of call-type queues. When an agent from this group becomes available, it selects the first nonempty queue from its list and picks the first call in the queue.

EXAMPLE 1. In our first example, there are five call types and 12 skill groups, defined by the rows and columns of Table 1, respectively. The priorities are very simple: an arriving call checks for an available agent by numeric order of agent groups and an agent that becomes available also looks at call queues in numeric order. Such simplistic priorities make the

Table 1 Skill Groups for Example 1

k	Agent group i											
1	1		3	4	5		7	8	9		11	12
2			3			6	7	8			11	12
3		2		4		6	7		9	10	11	12
4					5					10		12
5								8	9	10	11	12

Table 2 Results for Example 1

Aban.	l_k	Algo.	Cuts	Subgrad.	Sim.	CPU (sec.)	Obj.	QoS	QoS-KT	QoS per call type	Staffing vector
No	0.0	IP	9	9	120	116	220.9	0.804	0.115	(0.99, 0.90, 0.99, 0.80, 0.00)	(4, 36, 5, 26, 45, 45, 1, 13, 0, 24, 0, 0)
No	0.0	LP	6	6	70	66	222.1	0.803	0.454	(0.99, 0.86, 0.99, 0.84, 0.00)	(36, 38, 0, 0, 45, 46, 2, 15, 0, 23, 0, 0)
Yes	0.0	IP	10	10	129	114	219.5	0.801		(0.99, 0.93, 0.95, 0.84, 0.21)	(32, 30, 0, 1, 48, 44, 0, 30, 0, 16, 0, 0)
Yes	0.0	LP	4	4	57	45	220.5	0.804		(0.99, 0.94, 0.97, 0.88, 0.12)	(36, 30, 15, 0, 45, 39, 0, 17, 0, 21, 0, 0)
No	0.5	IP	9	5	73	67	263.2	0.863	0.860	(0.99, 0.88, 0.99, 0.92, 0.59)	(0, 30, 0, 24, 0, 0, 0, 0, 0, 65, 0, 92)
No	0.5	LP	7	3	60	50	251.9	0.803	0.866	(0.96, 0.51, 1.00, 0.94, 0.62)	(0, 47, 0, 13, 0, 14, 0, 66, 0, 80, 0, 0)
Yes	0.5	IP	35	29	365	280	224.7	0.801		(0.99, 0.61, 0.99, 0.85, 0.57)	(26, 25, 11, 1, 36, 39, 0, 0, 30, 35, 0, 0)
Yes	0.5	LP	14	13	167	131	225.4	0.809		(0.99, 0.76, 0.98, 0.76, 0.54)	(25, 24, 0, 7, 45, 44, 0, 12, 33, 14, 0, 0)

system highly unbalanced and the problem is then more difficult to solve because the service level of the low-priority call types tends to be very low (in the nonconcavity zone).

The cost of an agent with $s+1$ skills is assumed to be $1+\kappa s$, where $\kappa = 0.10$. The arrival rates and service rates per hour are $\lambda_1 = \lambda_3 = \lambda_5 = 440$, $\lambda_2 = \lambda_4 = 540$, and $\mu_k = 12$ for all k. We require the overall service level to be at least $l = 0.8$. For the values of l_k, we consider two cases: $l_k = 0$ (no constraint on the service level per call) and $l_k = 0.5$. Regarding the abandonments, we also consider two cases: (1) $\nu_k = 0$ (no abandonments), and (2) $\nu_k = 10$ for each k. This gives a total of four combinations. For each of them, we use the original method that solves an IP at each step and the method that replaces this IP by an LP and round up every solution. The results for the eight cases are in Table 2. For each case, the table indicates if there were abandonments (aban.), gives the minimal service level per call type ($l_k = 0.0$ or 0.5), says if the IP or the LP was solved at each step (algo.), then gives the total number of cuts generated by the algorithm excluding the initial load-covering constraints (cuts), the number of points $\mathbf{y}$ where subgradients (global or per call type) were estimated (subgrad.), the total number of simulations (sim.), the total CPU time in seconds (CPU), the global service level for the final solution (QoS), the global service level estimated by the Koole-Talim approximation (Koole et al. 2003) for that same solution (QoS-KT), the service level per call type (QoS per call type) for the final solution, and the staffing vector $\mathbf{y}$ in the final solution. The service-level values reported here are accurate to approximately ±0.02% with 95% confidence. The Koole-Talim approximation can be applied only when there are no abandonments. Roughly, we need $I+1 = 13$ 50-hour simulations each time we compute a subgradient. The number of times we compute a subgradient is typically equal to the number of generated cuts when $l_k = 0$ and smaller when $l_k > 0$ because in that case several cuts can be added at the same time.

Except for one case (see below), solving the IP as an LP at each step (i.e., using the rounding method) increases the value of the objective function for the final solution, by about 1% on average, while reducing the CPU time roughly by a factor of two or three. For a small example such as this one, it may be worth spending this additional CPU time to get a better solution, i.e., solve the IP at each iteration. For larger examples, however, the factor of CPU times between the IP and LP cases can be huge, and solving the LP eventually becomes the only practical solution. The exception is the case where $l_k = 0.5$ and there are no abandonments. Then, the IP and LP methods give totally different staffing vectors, with a large difference in the objective function value, and the LP happens to find a better solution than the IP. We reran this particular (difficult) case with 500 hours of simulation and the IP method found a much better solution, with value of 244.3, whereas LP found one with value 248.2.

Another interesting observation is that when $l_k = 0$, call type 5 has a very poor service level in the optimal solution: only a small fraction are answered within 20 seconds when there are abandonments and none when there are no abandonments. In the no-abandonments case, the system is actually unstable for call type 5. An infinite queue does build up with the retained solution, even if there is enough skill supply to cover more than the load of each call type simultaneously. (This explains the fact that the average of service levels per call type weighted by their arrival rates does not match the global service level; several calls of type 5 are still in the queue at the end of the simulation and are not accounted for.) The instability is due to the fact that each agent group that can handle calls of type 5 considers them with the lowest possible priority, and can handle at least two other types of calls with higher priorities. With these fixed (and unbalanced) priority rules, when we add agents to these groups, these agents first take care of other call types before handling calls of type 5 and there is not enough capacity left to handle their load. So, we must add a large number of these (rather generalist) agents to really increase the service level of this call type. This is why the objective function

increases significantly when we impose a minimal service level per call type with $l_k = 0.5$. This increase is much more pronounced when there are no abandonments because the abandonments (of all call types) during the periods of high congestion have a large impact in improving the service level of call type 5. For the case where $l_k = 0.5$ with abandonments, with the retained solutions with both IP and LP, the abandonment ratio is approximately 3.5% globally and 12% for call type 5. Of course, less expensive solutions could be found if we were allowed to change the priority rules of the agents.

When solving the case $l_k = 0.5$ without abandonments, we first add cuts only for the global constraint (as described in §3.3) and come up with a solution with a service level of zero for call type 5. At that point, we cannot add a subgradient cut based on the service-level constraint of this call type because the subgradient is zero. This is a case where we must inflate α_5 in the max-flow problem.

The Koole-Talim approximation for the global service level gives totally unreliable values when the system is highly unbalanced and some call types have very low service levels. It behaves much better for balanced systems. We also observed this in other examples and this had been pointed out already by Koole et al. (2003).

EXAMPLE 2. This is a slightly larger example, with 20 call types and 15 agent groups, defined by the rows and columns of Table 3. The priority rules, cost function, abandonment rates, and service-level requirements are as in the first example. The arrival rates and service rates per hour are $\lambda_1 = \lambda_2 = 240$, $\lambda_3 = \lambda_{17} = 160$, $\lambda_4 = \lambda_7 = \lambda_9 = \lambda_{14} = \lambda_{19} = \lambda_{20} = 260$, $\lambda_5 = \lambda_8 = \lambda_{16} = \lambda_{18} = 130$, $\lambda_6 = \lambda_{15} = 230$, $\lambda_{10} = 125$, $\lambda_{11} = 235$, $\lambda_{12} = 155$, $\lambda_{13} = 225$, and $\mu_k = 12$ for all k. The results are in Table 4.

Table 3 Skill Groups for Example 2

k	Agent group i														
1	1		3		5		7		9						
2	1										11		13		15
3		2		4		6		8		10					
4				4								12		14	
5	1	2			5						11				
6			3				7	8		10					
7					5				9			12	13		
8					5	6				10		12		14	15
9		2		4	5	6				10					
10					5	6			9				13	14	
11	1				5			8		10		12			
12				4					9		11			14	15
13		2			5		7			10					15
14			3					8	9				13		15
15		2				6	7							14	
16	1				5					10		12			
17		2				6					11				
18			3	4									13	14	
19		2						8				12			15
20			3			6		8				12	13	14	

Again, the rounding method increases the objective function value by about 1% on average, and reduces the CPU time by a factor of one or four. The unbalanced priority rules bring the same types of problems as in Example 1. The low-priority calls get extremely poor service level with $l_k = 0$ and the objective function value increases significantly when we impose $l_k = 0.5$ and there are no abandonments. For $l_k = 0.5$ with abandonments, for the retained solutions with IP and LP, the abandonment ratio is approximately 3.5% globally and higher for the low-priority call types. For example, for call types 16, 18, and 20, it is 9%, 10%, and 14%, respectively.

4.2. Staffing a Large Center

We now consider a model inspired by a large real-life call center operated by Bell Canada. There are 89 call types and 65 skill groups. We do not provide the details of the arrival, service, and abandon rates, and the call push/pull matrices because of space limitation and the proprietary nature of some of these details. In summary, the model assumes that the arrivals are Poisson with rates λ_k varying from 1.046 to 416.6, and the global rate is 3,581.7. The service times are assumed exponential with rates μ_k varying from 0.6777 to 600. The aggregate load ($\sum_k \lambda_k/\mu_k$) is 500. The patience times are exponential, with $\nu_k = 2.0$ for all k. For the service-level constraints, we take $l = 0.8$ globally and $l_k = 0.5$ for each call type.

Both the call types and the agent groups are split between two locations. One location has 22 call types and 15 agent groups; the second location has 43 call types and 74 agent groups. The number of skills per agent group goes from 1 to 24. We model the cost of an agent with $s+1$ skills as $1 + \kappa s$, where $\kappa = 0.05$.

The router uses a set of priority rules named "local specialist routing policy," which tries to assign any incoming call primarily to an agent based in the location from where the call originates. If more than one such local agent can handle this call, the router chooses the agent with the smallest number of skills. In case of equalities, the agent with the longest idle time is chosen. If no local agent is available to serve the call, it is put in a queue corresponding to its type. When a call has spent more than six seconds in the queue, the router tries to assign it to any agent from the other location. If the router succeeds, the call leaves the queue and is served remotely. On the other hand, when an agent becomes free, all the local queues are queried for a call and the call with the longest waiting time is taken. By using this policy, the router behaves as if there was only one queue for all call types. If there are still free agents after the queues

Table 4 Results for Example 2

Aban.	l_i	Algo.	Cuts	Subgrad.	Sim.	CPU (sec.)	Obj.	QoS	QoS-KT	QoS per call type	Staffing vector
No	0.0	IP	10	10	159	232	467.7	0.802	0.079	(1.00, 0.99, 0.99, 0.93, 0.98, 0.99, 0.76, 0.00, 0.61, 0.30, 0.82, 0.97, 0.98, 0.94, 0.94, 0.27,0.88, 0.89, 0.00, 0.46)	(31, 0, 55, 61, 0, 0, 74, 21, 33, 0, 54, 12, 0, 0, 0,)
No	0.0	LP	2	2	40	62	472.6	0.818	0.280	(1.00, 0.99, 0.99, 0.96, 0.99, 0.99, 0.94, 0.00, 0.76, 0.77, 0.86, 0.99, 0.98, 0.90, 0.93, 0.37, 0.97, 0.81, 0.00, 0.00)	(30, 0, 27, 68, 0, 0, 78, 22, 50, 0, 57, 11, 0, 0, 1)
Yes	0.0	IP	7	7	125	184	466.6	0.800		(1.00, 1.00, 0.99, 0.95, 0.99, 0.99, 0.93, 0.34, 0.82, 0.80, 0.85, 0.99, 0.98, 0.91, 0.94, 0.48, 0.96, 0.85, 0.03, 0.07)	(32, 0, 23, 72, 0, 0, 77, 17, 52, 0, 51, 0, 0, 1, 15)
Yes	0.0	LP	7	7	114	164	470.8	0.807		(1.00, 0.99, 0.99, 0.95, 0.99, 0.99, 0.84, 0.15, 0.83, 0.63, 0.87, 0.99, 0.99, 0.94, 0.96, 0.45, 0.95, 0.92, 0.03, 0.39)	(30, 0, 38, 72, 0, 0, 77, 22, 43, 0, 50, 0, 0, 0, 11)
No	0.5	IP	40	23	362	486	501.6	0.872	0.799	(1.00, 0.98, 0.99, 0.99, 0.96, 0.99, 0.94, 0.50, 0.91, 0.78, 0.98, 0.96, 0.86, 0.98, 0.57, 0.75, 0.56, 0.97, 0.69, 0.78)	(46, 9, 52, 63, 0, 0, 42, 51, 42, 0, 27, 13, 0, 10, 0)
No	0.5	LP	26	9	158	226	500.4	0.868	0.805	(1.00,0.99, 0.99, 0.96, 0.97, 0.99, 0.98, 0.50, 0.64, 0.90, 0.98, 0.98, 0.90, 0.98, 0.61, 0.76, 0.79, 0.90, 0.71, 0.70)	(41, 0, 41, 55, 0, 0, 51, 53, 56, 0, 36, 16, 0, 0, 8)
Yes	0.5	IP	9	3	57	105	474.0	0.831		(1.00, 0.99, 0.99, 0.90, 0.97, 0.99, 0.76, 0.57, 0.62, 0.50, 0.98, 0.95, 0.98, 0.95, 0.89, 0.55, 0.78, 0.52, 0.76, 0.51)	(29, 0, 9, 59, 0, 0, 69, 72, 40, 0, 41, 0, 0, 0, 20)
Yes	0.5	LP	12	3	54	87	473.4	0.825		(1.00, 1.00, 0.97, 0.83, 0.99, 0.99, 0.79, 0.91, 0.50, 0.53, 0.77, 0.99, 0.99, 0.97, 0.94, 0.53, 0.88, 0.88, 0.50, 0.51)	(43, 0, 45, 61, 0, 0, 69, 0, 40, 0, 42, 0, 0, 0, 43)

are queried, the query is made a second time, allowing agents to serve calls from the other location.

We first tried to solve this problem as an IP at each step, but the CPU time requirement was unreasonably high. So, we solved the LP (with rounding up) instead. The resolution took 37 minutes of CPU time. Less than 1% of this time was used for solving the LPs; most of it was used for the simulations. Finding the initial solution that satisfies the load-coverage constraints took six seconds and generated 82 constraints.

We obtained a solution with 526 agents (for a load of 500), for which the objective function value was 669.35. The average number of skills per agent is 6.45, and the retained staffing vector has 37 nonzero entries ranging from 1 to 68. For that solution, the 50-hour simulation (used by our algorithm) estimates a global service level at 0.83 ± 0.02 (with 95% confidence), the average agent occupancy at 94.5%, and the abandonment ratio at 1.1%. We also ran a 5,000-hour simulation with this retained solution to check the service levels with better precision. This time, the global service level was estimated as 0.845 ± 0.002, and three call types had estimated service levels slightly below 0.5, namely, 0.49 ± 0.01, 0.47 ± 0.01, and 0.47 ± 0.01 (with 95% confidence). After observing this, it would be easy to add a few more cuts to increase these three service levels and do one or two additional iterations, but we did not do that.

Interestingly, the five call types having the largest arrival rates, which account for 42.3% of the total call volume, are also those that have the largest service level, all around 0.99, in the retained solution. The explanation is that these call types are favored by the current (fixed) priority rules. In contrast, the sixth most-frequent call type has its service level at the lower bound of 0.5, with an abandonment ratio of 5.2%. This call type can be handled by a single-agent group. This group has 53 agents with 11 different skills and is at a different location than the call type, which explains the poor service level. Its occupation rate is 93.7%, a little below the average.

It must be emphasized that the retained solution is random and may change significantly when we vary the sample size and/or the random number streams. To illustrate this, we ran the same algorithm

with 500-hour instead of 50-hour simulations, and found a solution with objective function value of 664.3 after about six hours of CPU time. We also replicated the algorithm five more times independently with 50-hour simulations and obtained solutions whose values ranged from 663.1 to 693.3, with global QoS ranging from 0.84 to 0.89, and CPU times ranging from 31 to 59 minutes.

In this model, the objective function could be reduced significantly by reducing the average number of skills per agent and optimizing the skill mixes, agent group locations, and routing rules and priorities. This is beyond the scope of this paper and will be addressed in future work.

5. Conclusion

We have adapted the cutting-plane methodology of Atlason et al. (2004) to the problem of optimal staffing of a multiskill call center. We introduced several heuristics to make the approach practical for large problem instances. These heuristics include, for example, getting an initial set of constraints by solving a max-flow problem, computing finite differences with steps larger than one (selected adaptively), and rounding up the solution of an LP instead of solving the exact IP. The methodology replaces the unknown expectations in the constraints of the original problem by sample averages. It is a heuristic: there is no guarantee that the optimal solution to the sample problem is optimal or even feasible for the original problem, and no guarantee that the returned solution is optimal or feasible for the sample problem. But "good" (not too far from optimal) solutions are returned most of the time. These solutions can (and should) be refined at the end of the algorithm via local search heuristics. Further work is needed in that direction. It is also a good idea to run the algorithm more than once with different streams of random numbers and perhaps slightly different parameters, and pick up the best of the solutions thus obtained. This provides a viable approach to obtain reasonable solutions for important practical problems for which no methodology that we know can return guaranteed optimal solutions.

6. Electronic Companion

An electronic companion to this paper is available as part of the online version that can be found at http://mansci.journal.informs.org/.

Acknowledgments

This research was supported by Grants OGP-0110050 and CRDPJ-251320 from NSERC-Canada, a grant from Bell Canada (via the "Laboratoires Universitaires Bell"), Grant 02ER3218 from FQRNT-Québec, and a Canada Research Chair to the second author. The authors thank Éric Buist, who implemented the call center simulation package, Wyean Chan, who helped with the programing, and Athanassios Avramidis, who gave several insightful comments.

References

Atlason, J., M. A. Epelman, S. G. Henderson. 2003. Using simulation to approximate subgradients of convex performance measures in service systems. *Proc. 2003 Winter Simulation Conf.*, IEEE Press, Piscataway, NJ, 1824–1832.

Atlason, J., M. A. Epelman, S. G. Henderson. 2004. Call center staffing with simulation and cutting plane methods. *Ann. Oper. Res.* **127** 333–358.

Avramidis, A. N., A. Deslauriers, P. L'Ecuyer. 2004. Modeling daily arrivals to a telephone call center. *Management Sci.* **50**(7) 896–908.

Buist, E., P. L'Ecuyer. 2005. A java library for simulating contact centers. *Proc. 2005 Winter Simulation Conf.*, IEEE Press, Piscataway, NJ, 556–565.

Ellis, R. S. 1985. *Entropy, Large Deviations, and Statistical Mechanics*. Springer Verlag, New York.

Gans, N., G. Koole, A. Mandelbaum. 2003. Telephone call centers: Tutorial, review, and research prospects. *Manufacturing Service Oper. Management* **5**(2) 79–141.

Garnett, O., A. Mandelbaum. 2000. An introduction to skills-based routing and its operational complexities. Teaching note, Technion University, Israel.

Henderson, S., A. Mason. 1998. Rostering by iterating integer programming and simulation. *Proc. 1998 Winter Simulation Conf.*, Vol. 1, IEEE Press, Piscataway, NJ, 677–683.

Ingolfsson, A., E. Cabral, X. Wu. 2003. Combining integer programming and the randomization method to schedule employees. Technical report, School of Business, University of Alberta, Edmonton, Alberta, Canada.

Jagers, A. A., E. A. van Doorn. 1991. Convexity of functions which are generalizations of the Erlang loss function and the Erlang delay function. *SIAM Rev.* **33** 281–282.

Jennings, O. B., A. Mandelbaum, W. A. Massey, W. Whitt. 1996. Server staffing to meet time-varying demand. *Management Sci.* **42**(10) 1383–1394.

Kelley Jr., J. E. 1960. The cutting-plane method for solving convex programs. *J. Soc. Indust. Appl. Math.* **8**(4) 703–712.

Koole, G., A. Mandelbaum. 2002. Queueing models of call centers: An introduction. *Ann. Oper. Res.* **113** 41–59.

Koole, G., J. Talim. 2000. Exponential approximation of multi-skill call centers architecture. *Proc. QNETs*, Ilkley, UK, 23/1–10.

Koole, G., A. Pot, J. Talim. 2003. Routing heuristics for multi-skill call centers. *Proc. 2003 Winter Simulation Conf.*, IEEE Press, Piscataway, NJ, 1813–1816.

Law, A. M., W. D. Kelton. 2000. *Simulation Modeling and Analysis*, 3rd ed. McGraw-Hill, New York.

L'Ecuyer, P. 2004. *SSJ: A Java Library for Stochastic Simulation*. Software user's guide. http://www.iro.umontreal.ca/~lecuyer.

L'Ecuyer, P., R. Simard, E. J. Chen, W. D. Kelton. 2002. An object-oriented random-number package with many long streams and substreams. *Oper. Res.* **50**(6) 1073–1075.

Mandelbaum, A., M. I. Reimann. 1998. On pooling in queueing networks. *Management Sci.* **44**(7) 971–981.

Shwartz, A., A. Weiss. 1995. *Large Deviations for Performance Analysis*. Chapman and Hall, London.

Vogel, S. 1994. A stochastic approach to stability in stochastic programming. *J. Comput. Appl. Math.* **56** 65–96.

Wallace, R. B., W. Whitt. 2005. A staffing algorithm for call centers with skill-based routing. *Manufacturing Service Oper. Management* **7**(4) 276–294.

Yakowitz, S., P. L'Ecuyer, F. Vázquez-Abad. 2000. Global stochastic optimization with low-discrepancy point sets. *Oper. Res.* **48**(6) 939–950.

Optimizing Daily Agent Scheduling in a Multiskill Call Center

Athanassios N. Avramidis
School of Mathematics, University of Southampton
Highfield, Southampton, SO17 1BJ, UNITED KINGDOM

Michel Gendreau
Département d'informatique et de recherche opérationnelle and CIRRELT
Université de Montréal, C.P. 6128, Succ. Centre-Ville
Montréal (Québec), H3C 3J7, CANADA

Pierre L'Ecuyer
Département d'informatique et de recherche opérationnelle, CIRRELT and GERAD
Université de Montréal, C.P. 6128, Succ. Centre-Ville
Montréal (Québec), H3C 3J7, CANADA

Ornella Pisacane
Dipartimento di Elettronica Informatica e Sistemistica
Università della Calabria, Via P. Bucci, 41C
Arcavacata di Rende (CS), ITALY

November 13, 2007

Abstract

We examine and compare simulation-based algorithms for solving the agent scheduling problem in a multiskill call center. This problem consists in minimizing the total costs of agents under constraints on the expected service level per call type, per period, and aggregated. We propose a solution approach that combines simulation with integer or linear programming, with cut generation. In our numerical experiments with realistic problem instances, this approach performs better than all other methods proposed previously for this problem. We also show that the two-step approach, which is the standard method for solving this problem, sometimes yield solutions that are highly suboptimal and inferior to those obtained by our proposed method.

1 Introduction

The telephone call center industry employs millions of people around the world and is fast growing. In the United States, for example, customer service representatives held 2.1 million jobs in 2004, and employment in this job category is expected to increase faster than average at least through 2014 (Bureau of Labor Statistics 2007). A few percent saving in workforce salaries easily means several million dollars.

Call centers often handle several types of calls distinguished by the required skills for delivering service. Training all agents to handle all call types is not cost-effective. Each agent has a selected number of skills and the agents are distinguished by the set of call types they can handle (also called their *skill set*). When such skill constraints exist, we speak of a *multiskill* call center. *Skill-based routing* (SBR), or simply *routing*, refers to the rules that control the call-to-agent and agent-to-call assignments. Most modern call centers perform skill-based routing (Koole and Mandelbaum 2002, Gans et al. 2003).

In a typical call center, inbound calls arrive at random according to some complicated stochastic processes, call durations are also random, waiting calls may abandon after a random patience time, some agents may fail to show up to work for any reason, and so on. Based on forecasts of call volumes, call center managers must decide (among other things) how many agents of each type (i.e., skill set) to have in the center at each time of the day, must construct working schedules for the available agents, and must decide on the call routing rules. These decisions are made under a high level of uncertainty. The goal is typically to provide the required quality of service at minimal cost.

The most common measure of quality of service is the *service level* (SL), defined as the long-term fraction of calls whose time in queue is no larger than a given threshold. Frequently, multiple measures of SL are of interest: for a given time period of the day, for a given call type, for a given combination of call type and period, aggregated over the whole day and all call types, and so on. For certain call centers that provide public services, SL constraints are imposed by external authorities, and violations may result in stiff penalties (CRTC 2000).

In this paper, we assume that we have a detailed stochastic model of the dynamics of the call center for one day of operation. This model specifies the stochastic processes for the call arrivals (these processes are usually non-stationary and doubly stochastic), the distributions of service times and patience times for calls, the call routing rules, the periods of unavailability of agents between calls (e.g., to fill out forms, or to go to the restroom, etc.), and so forth. We formulate a stochastic optimization problem where the objective is to minimize the total cost of agents, under various SL constraints. This could be used in long-term planning, to decide how many agents to hire and for what skills to train them, or for short-term planning, to decide which agents to call for work on a given day and what would be their work schedule. The problem is difficult because for any given fixed staffing of agents (the staffing determines how many agents of each type are available in each

time period), no reliable formulas or quick numerical algorithms are available to estimate the SL; it can be estimated accurately only by long (stochastic) simulations. Scheduling problems are in general NP-hard, even in deterministic settings where each solution can be evaluated quickly and exactly. When this evaluation requires costly and noisy simulations, as is the case here, solving the problem exactly is even more difficult and we must settle with methods that are partly heuristic.

Staffing in the *single-skill* case (i.e., single call type and single agent type) has received much attention in the call center literature. Typically, the workload varies considerably during the day (Gans et al. 2003, Avramidis et al. 2004, Brown et al. 2005), and the planned staffing can change only at a few discrete points in time (e.g., at the half hours). It is common to divide the day into several periods during which the staffing is held constant and the arrival rate does not vary much. If the system can be assumed to reach steady-state quickly (relative to the length of the periods), then steady-state queueing models are likely to provide a reasonably good staffing recommendation for each period. For instance, in the presence of abandonments, one can use an Erlang-A formula to determine the minimal number of agents for the required SL in each period (Gans et al. 2003). When that number is large, it is often approximated by the *square root safety staffing formula*, based on the Halfin-Whitt heavy-traffic regime, and which says roughly that the capacity of the system should be equal to the workload plus some safety staffing which is proportional to the square root of the workload (Halfin and Whitt 1981, Gans et al. 2003). This commonly used heuristic, known as the stationary independent period by period (SIPP) approach, often fails to meet target SL because it neglects the non-stationarity (Green et al. 2003). Non-stationary versions of these approximations have also been developed, still for the single-skill case (Jennings et al. 1996, Green et al. 2003).

Scheduling problems are often solved in two separate steps (Mehrotra 1997): After an appropriate staffing has been determined for each period in the first step, a minimum-cost set of shifts that covers this staffing requirement can be computed in the second step by solving a linear integer program. However, the constraints on admissible working shifts often force the second step solution to overstaff in some of the periods. This drawback of the two-step approach has been pointed out by several authors, who also proposed alternatives (Keith 1979, Thompson 1997, Henderson and Mason 1998, Ingolfsson et al. 2003, Atlason et al. 2004). For example, the SL constraint is often only for the time-aggregated (average) SL over the entire day; in that case, one may often obtain a lower-cost scheduling solution by reducing the minimal staffing in one period and increasing it in another period. Atlason et al. (2004) developed a *simulation-based* methodology to optimize agents' scheduling in the presence of uncertainty and general SL constraints, based on simulation and cutting-plane ideas. Linear inequalities (cuts) are added to an integer program until its optimal solution satisfies the required SL constraints. The SL and the cuts are estimated by simulation.

In the *multiskill case*, the staffing and scheduling problems are more challenging, because the workload can be covered by several possible combinations of skill sets, and the routing rules also have a strong impact on the performance. Staffing a single period in steady-state is already difficult;

the Erlang formulas and their approximations (for the SL) no longer apply.. Simulation seems to be the only reliable tool to estimate the SL. Cezik and L'Ecuyer (2007) adapt the simulation-based methodology of Atlason et al. (2004) to the *optimal staffing* of a multiskill call center for a *single period*. They point out difficulties that arise with this methodology and develop heuristics to handle them. Avramidis et al. (2006) solve the same problem by using neighborhood search methods combined with an analytical approximation of SLs, with local improvement via simulation at the end. Pot et al. (2007) impose a constraint only on the aggregate SL (across all call types); they solve Lagrangean relaxations using search methods and analytical approximations.

Some authors have studied the special case where there are only two call types, and some have developed queueing approximations for the case of two call types, via Markov chains and under simplifying assumptions; see Stolletz and Helber (2004) for example. But here we are thinking of 20 to 50 call types or more, which is common in modern call centers, and for which computation via these types of Markov chain models is clearly impractical.

For the *multiskill scheduling problem*, Bhulai et al. (2007) propose a two-step approach in which the first step determines a staffing of each agent type for each period, and the second step computes a schedule by solving an IP in which this staffing is the right-hand side in key constraints. A key feature of the IP model is that the staff-coverage constraints allow *downgrading* an agent into any alternative agent type with smaller skill set, separately for each period. Bhulai et al. (2007) recognize that their two-step approach is generally suboptimal and they illustrate this by examples.

In this paper, we propose a simulation-based algorithm for solving the multiskill scheduling problem, and compare it to the approach of Bhulai et al. (2007). This algorithm extends the method of Cezik and L'Ecuyer (2007), which solves a single-period staffing problem. In contrast to the two-step approach, our method optimizes the staffing and the scheduling simultaneously. Our numerical experiments show that our algorithm provides approximate solutions to large-scale realistic problem instances in reasonable time (a few hours). These solutions are typically better, sometimes by a large margin (depending on the problem), than the best solutions from the two-step approach. We are aware of no competitive faster method.

The remainder of this paper is organized as follows. In section 2, we formally define the problem at hand and provide a mathematical programming formulation. The new algorithm is described in 3. We report computational results on several test instances in section 4. The conclusion follows. A preliminary version of this paper was presented at the 2007 Industrial Simulation Conference (Avramidis et al. 2007a).

2 Model Formulation

We now provide definitions of the multiskill staffing and scheduling problems. We assume that we have a stochastic model of the call center, under which the mathematical expectations used below

are well defined, and that we can simulate the dynamics of the center under this model. Our problem formulations here do not depend on the details of this model.

There are K call types, labeled from 1 to K, and I agent types, labeled from 1 to I. Agent type i has the skill set $S_i \subseteq \{1,\ldots,K\}$. The day is divided into P periods of given length, labeled from 1 to P. The *staffing vector* is $\mathbf{y} = (y_{1,1},\ldots,y_{1,P},\ldots,y_{I,1},\ldots,y_{I,P})^{\mathrm{t}}$ where $y_{i,p}$ is the number of agents of type i available in period p. Given $\mathbf{y}$, the *service level* (SL) in period p for type-k calls is defined as

$$g_{k,p}(\mathbf{y}) = \mathbb{E}[S_{\mathrm{g},k,p}]/\mathbb{E}[S_{k,p} + A_{k,p}],$$

where $S_{k,p}$ is the number of type-k calls that arrive in period p, $S_{\mathrm{g},k,p}$ is the number of those calls that get served after waiting at most $\tau_{k,p}$ (a constant called the *acceptable waiting time*), and $A_{k,p}$ is the number of type-k calls that abandon in period p after waiting at least $\tau_{k,p}$. Aggregate SLs, per call type, per period, and globally, are defined analogously. Given acceptable waiting times τ_p, τ_k, and τ, the aggregate SLs are denoted by $g_p(\mathbf{y})$, $g_k(\mathbf{y})$ and $g(\mathbf{y})$ for period p, call type k, and overall, respectively.

A *shift* is a time pattern that specifies the periods in which an agent is available to handle calls. In practice, it is characterized by its *start period* (the period in which the agent starts working), *break periods* (the periods when the agent stops working), and *end period* (the period when the agent finishes his/her workday). In general, agents have several breaks of different duration; for instance, morning and afternoon coffee breaks, as well as a longer lunch break.

Let $\{1,\ldots,Q\}$ be the set of all admissible shifts. To simplify the exposition, we assume that this set is the same for all agent types; this assumption could easily be relaxed if needed, by introducing specific shift sets for each agent type. The admissible shifts are specified via a $P \times Q$ matrix $\mathbf{A}_0$ whose element (p,q) is $a_{p,q} = 1$ if an agent with shift q works in period p, and 0 otherwise. A vector $\mathbf{x} = (x_{1,1},\ldots,x_{1,Q},\ldots,x_{I,1},\ldots,x_{I,Q})^{\mathrm{t}}$, where $x_{i,q}$ is the number of agents of type i working shift q, is a *schedule*. The *cost vector* is $\mathbf{c} = (c_{1,1},\ldots,c_{1,Q},\ldots,c_{I,1},\ldots,c_{I,Q})^{\mathrm{t}}$, where $c_{i,q}$ is the cost of an agent of type i with shift q. To any given shift vector $\mathbf{x}$, there corresponds the staffing vector $\mathbf{y} = \mathbf{A}\mathbf{x}$, where $\mathbf{A}$ is a block-diagonal matrix with I identical blocks $\mathbf{A}_0$, if we assume that each agent of type i works as a type-i agent for his/her entire shift.

However, following Bhulai et al. (2007), we also allow an agent of type i to be downgraded to an agent with smaller skill set, i.e., of type i_p where $S_{i_p} \subset S_i$, in any time period p of his/her shift. Define $\mathcal{S}_i^+ = \{j : S_j \supset S_i \wedge \nexists m : S_j \supset S_m \supset S_i\}$ ($\mathcal{S}_i^+$ is thus the set of agent types whose skill set is a minimum strict superset of the skill set of agent type i) and $\mathcal{S}_i^- = \{j : S_j \subset S_i \wedge \nexists m : S_j \subset S_m \subset S_i\}$ ($\mathcal{S}_i^-$ is thus the set of agent types whose skill set is a maximum strict subset of skill set of agent type i). To illustrate, consider a call centre with $K = 3$ call types, $I = 4$ agent types, and skill sets $S_1 = \{1\}$, $S_2 = \{2\}$ (specialist agents), $S_3 = \{2,3\}$, and $S_4 = \{1,2,3\}$ (generalist agent); then we have, among others, $\mathcal{S}_1^- = \mathcal{S}_2^- = \emptyset$, $\mathcal{S}_2^+ = \{3\}$, and $\mathcal{S}_4^- = \{1,3\}$. For each i and $j \in \mathcal{S}_i^-$ and each period p, we define the *skill transfer* variable $z_{i,j,p}$, which represents the number of type-i agents that are downgraded to type j during period p. Note that by performing multiple skill transfers during

a period, an agent of type i may end up being downgraded to any type whose skill set included in S_i (in the previous example, a type 4 agent could be downgraded to type 3 and then to type 2, even though there are no $z_{4,2,p}$ variables).

A schedule $\mathbf{x} = (x_{1,1},\ldots,x_{1,Q},\ldots,x_{I,1},\ldots,x_{I,Q})^t$ is said to *cover the staffing* $\mathbf{y} = (y_{1,1},\ldots,y_{1,P},\ldots, y_{I,1},\ldots,y_{I,P})^t$ if for $i = 1,\ldots,I$ and $p = 1,\ldots,P$, there are nonnegative integers $z_{j,i,p}$ for $j \in \mathcal{S}_i^+$ and $z_{i,j,p}$ for $j \in \mathcal{S}_i^-$, such that

$$\sum_{q=1}^{Q} a_{p,q}x_{i,q} + \sum_{j\in\mathcal{S}_i^+} z_{j,i,p} - \sum_{j\in\mathcal{S}_i^-} z_{i,j,p} \geq y_{i,p}. \tag{1}$$

These inequalities can be written in matrix form as $\mathbf{Ax} + \mathbf{Bz} \geq \mathbf{y}$, where $\mathbf{z}$ is a column vector whose elements are the $z_{i,j,p}$ variables and $\mathbf{B}$ is a matrix whose entries are in the set $\{-1,0,1\}$. With this notation, the *scheduling problem* can be formulated as

(P0) : [Scheduling problem]

$$\begin{aligned}
\min\ & \mathbf{c}^t\mathbf{x} = \textstyle\sum_{i=1}^{I}\sum_{q=1}^{Q} c_{i,q}x_{i,q} \\
\text{s.t.}\ & \\
& \mathbf{Ax} + \mathbf{Bz} \geq \mathbf{y} \\
& g_{k,p}(\mathbf{y}) \geq l_{k,p} \text{ for } 1 \leq k \leq K \text{ and } 1 \leq p \leq P \\
& g_p(\mathbf{y}) \geq l_p \quad \text{for } 1 \leq p \leq P \\
& g_k(\mathbf{y}) \geq l_k \quad \text{for } 1 \leq k \leq K \\
& g(\mathbf{y}) \geq l \\
& \mathbf{x} \geq 0,\ \mathbf{z} \geq 0,\ \mathbf{y} \geq 0 \text{ and integer}
\end{aligned}$$

where $l_{k,p}$, l_p, l_k and l are given constants.

In practice, a given agent often works more efficiently (faster) when handling a smaller number of call types (i.e., if his/her skill set is artificially reduced). The possibility of downgrading agents to a smaller skill set for some periods can sometimes be exploited to take advantage of this increased efficiency. In case where the agent's speed for a given call type (in the model) does not depend on his/her skill set, one might think intuitively that downgrading cannot help, because it only limits the flexibility of the routing. This would be true if we had an optimal dynamic routing of calls. But in practice, an optimal dynamic routing is too complicated to compute and simpler routing rules are used instead. These simple rules are often static. Then, downgrading may sometimes help by effectively changing the routing rules. Clearly, the presence of skill transfer variables in (P0) cannot increase the optimal cost, it can only reduce it.

Suppose we consider a single period, say period p, and we replace $g_{k,p}(\mathbf{y})$ and $g_p(\mathbf{y})$ by approximations that depend on the staffing of period p only, say $\tilde{g}_{k,p}(y_{1,p},\ldots,y_{I,p})$ and $\tilde{g}_p(y_{1,p},\ldots,y_{I,p})$, respectively. If all system parameters are assumed constant over period p, then natural approxima-

tions are obtained by assuming that the system is in steady-state over this period. The single-period multiskill staffing problems can then be written as

(P1) : [Staffing problem]

$$\begin{aligned} \min \quad & \textstyle\sum_{i=1}^{I} c_i y_i \\ \text{s.t.} \quad & \\ & \tilde{g}_k(y_1,\ldots,y_I) \geq l_k \quad \text{for } 1 \leq k \leq K \\ & \tilde{g}(y_1,\ldots,y_I) \geq l \\ & y_i \geq 0 \quad \text{and integer for all } i \end{aligned}$$

where c_i is the cost of agent type i (for a single period), and the period index was dropped throughout. Simulation-based solution methods for this problem are proposed in Cezik and L'Ecuyer (2007) and Avramidis et al. (2006). Pot et al. (2007) address a restricted version of it, with a single constraint on the aggregate SL over the period (i.e., they assume $l_k = 0$ for all k).

In the approach of Bhulai et al. (2007), the first step is to determine an appropriate staffing, $\hat{\mathbf{y}} = (\hat{y}_{1,1},\ldots,\hat{y}_{1,P},\ldots,\hat{y}_{I,1},\ldots,\hat{y}_{I,P})^{\mathrm{t}}$. For this, they look at each period p in isolation and solve a version of (P1) with a single constraint on the aggregate SL; this gives $\hat{y}_{1,p},\ldots,\hat{y}_{I,p}$ for each p. In their second step, they find a schedule that covers this staffing by solving:

(P2) : [Two-stage approach]

$$\begin{aligned} \min \quad & \mathbf{c}^{\mathrm{t}}\mathbf{x} \\ \text{s.t.} \quad & \\ & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} \geq \hat{\mathbf{y}} \\ & \mathbf{x} \geq 0, \mathbf{z} \geq 0 \quad \text{and integer} \end{aligned}$$

The presence of skill-transfer variables generally reduces the optimal cost in (P2) by adding flexibility, compared with the case where no downgrading is allowed. However, there sometimes remains a significant gap between the optimal solution of (P0) and the best solution found for the same problem by the two-step approach. The following simplified example illustrates this.

Example 1 Let $K = I = P = 3$, and $Q = 1$. The single type of shift covers the three periods. The skill sets are $S_1 = \{1,2\}$, $S_2 = \{1,3\}$, and $S_3 = \{2,3\}$. All agents have the same shift and the same cost. Suppose that the total arrival process is stationary Poisson with mean 100. This incoming load is equally distributed between call types $\{1,2\}$ in period 1, $\{1,3\}$ in period 2, $\{2,3\}$ in period 3. Any agent can be downgraded to a specialist that can handle a single call type (that belongs to his skill set), in any period. In the presence of such specialists, an incoming call goes first to its corresponding specialist if there is one available, otherwise it goes to a generalist that can handle

another call type as well. When the agent becomes available, he serves the call that has waited the longest among those in the queue (if any). The service times are exponential with mean 1, there are no abandonments, and the SL constraints specify that 80% of all calls must be served within 20 seconds, in each time period, on average over an infinite number of days.

If we assume that the system operates in steady-state in period 1, then the optimal staffing for that period is 104 agents of type 1. Since all agents can serve all calls, we have in this case an $M/M/s$ queue with $s = 104$, and the global SL is 83.4%, as can be computed by the Erlang-C formula. By symmetry, the optimal staffing solutions for the other periods are obviously the same: 104 agents of type 2 in period 2 and 104 agents of type 3 in period 3. Then, the two-step approach gives a solution to (P2) with 104 agents of each type, for a total of 312 agents.

Solving (P0) directly instead (e.g., using the simulation-based algorithm described in the next section), assuming again (as an approximation) that the system is in steady-state in each of the three periods, we find a feasible solution with 35 agents of type 1, 35 agents of type 2, and 34 agents of type 3, for a total of 104 agents. With this solution, during period 1, the agents of types 2 and 3 are downgraded to specialists who handle only call types 1 and 2, respectively, and the agents of type 1 act as generalists. A similar arrangement applies to the other periods, mutatis mutandis. Note that this solution of (P0) remains valid even if we remove the skill transfer variables from the formulation of (P0), because the sets Si^- and S_i^+ are all empty, if we assume that the routing rules do not change; i.e., if calls are always routed first to agents that can handle only this call type among the calls that can arrive during the current period.

Suppose now that we add the additional skill sets $S_4 = \{1\}$, $S_5 = \{2\}$, $S_6 = \{3\}$, and that these new specialists cost 6 each, whereas the agents with two skills cost 7. In this case it becomes attractive to use specialists to handle a large fraction of the load, because they are less expensive, and to keep a few generalists in each period to obtain a "resource sharing" effect. It turns out that an optimal staffing solution for period 1 is 2 generalists (type 1) and 52 specialists of each of the types 4 and 5. An analogous solution holds for each period. With these numbers, if downgrading is not possible, the two-step approach gives a solution with 6 generalists (2 of each type) and 156 specialists (52 of each type), for a total cost of 978. If downgrading is allowed, then the two-step approach finds the following much better solution: 2 agents of type 1 and 52 of each of the types 2 and 3, for a total cost of 742. The skill transfer works in this way. In period 1: 52 agents of type 2 are downgraded to specialists of type 4 and 52 of type 3 to specialists of type 5. In period 2: 2 agents of type 1 are downgraded to agents of type 5, 52 of type 2 to type 6 and 50 of type 3 to type 5. In period 3: 2 agents of type 1 are downgraded to agents of type 4, 50 of type 2 to type 4 and 52 of type 3 to type 6. If we solve (P0) directly with these additional skill sets, we get the same solution as without them; i.e., 104 agents with two skills each, for a total cost of 728. This is again better than with the two-step approach, but the gap is much smaller than what we had with only three skill sets.

Example 2 In the previous example, if all the load was from a single call type, there would be a single agent type and the two-step approach would provide exactly the same solution as the optimal solution of (P0). The example illustrates a suboptimality gap due to a variation in the *type* of load.

Another potential source of suboptimality (this one can occur even in the case of a single call type) is the time variation of the total load from period to period. If there is only a global SL constraint over the entire day, then the optimal solution may allow a lower SL during one (or more) peak period(s) and recover an acceptable global SL by catching up in the other periods. To account for this, Bhulai et al. (2007), Section 5.4, propose a heuristic based on the solution obtained by their basic two-step approach. Although this appears to work well in their examples, the effectiveness of this heuristic for general problems is not clear.

Yet another type of limitation that can significantly increase the total cost is the restriction on the set of available shifts.. Suppose for example that there is a single call type, that the day has 10 periods, and that all shifts must cover 8 periods, with 7 periods of work and a single period of lunch break after 3 or 4 periods of work. Thus a shift can start in period 1, 2, or 3, and there are six shift types in total. Suppose we need 100 agents available in each period. For this we clearly need 200 agents, each one working for 7 periods, for a total of 1400 agent-periods. If there were no constraints on the duration and shape of shifts, on the other hand, then 1000 agent-periods would suffice.

3 Optimization by Simulation and Cutting Planes

We now describe the proposed simulation-based optimization algorithm. The general idea is to replace the problem (P0) by a *sample* version of it, ($SP0_n$), and then replace the nonlinear SL constraints by a small set of linear constraints, in a way that the optimal solution of the resulting *relaxed* sample problem is close to that of (P0). The relaxed sample problem is solved by linear or integer programming.

We first describe how the relaxation works when applied directly to (P0); its works the same way when applied to the sample problem. Consider a version of (P0) in which the SL constraints have been replaced by a small set of linear constraints that do not cut out the optimal solution. Let $\bar{\mathbf{y}}$ be the optimal solution of this (current) relaxed problem. If $\bar{\mathbf{y}}$ satisfies all SL constraints of (P0), then it is an optimal solution of (P0) and we are done. Otherwise, take a violated constraint of (P0), say $g(\bar{\mathbf{y}}) < l$, suppose that g is (jointly) concave in $\mathbf{y}$ for $\mathbf{y} \geq \bar{\mathbf{y}}$, and that $\bar{\mathbf{q}}$ is a *subgradient* of g at $\bar{\mathbf{y}}$. Then

$$g(\mathbf{y}) \leq g(\bar{\mathbf{y}}) + \bar{\mathbf{q}}^t(\mathbf{y} - \bar{\mathbf{y}})$$

for all $\mathbf{y} \geq \bar{\mathbf{y}}$. We want $g(\mathbf{y}) \geq l$, so we must have

$$l \leq g(\mathbf{y}) \leq g(\bar{\mathbf{y}}) + \bar{\mathbf{q}}^t(\mathbf{y} - \bar{\mathbf{y}}),$$

i.e.,

$$\bar{\mathbf{q}}^t\mathbf{y} \geq \bar{\mathbf{q}}^t\bar{\mathbf{y}} + l - g(\bar{\mathbf{y}}). \tag{2}$$

Adding this linear *cut inequality* to the constraints removes $\bar{\mathbf{y}}$ from the current set of feasible solutions of the relaxed problem without removing any feasible solution of (P0). On the other hand, in case $\bar{\mathbf{q}}$ is not really a subgradient (which may happens in practice), then we may cut out feasible solutions of (P0), including the optimal one. We will return to this.

Since we cannot evaluate the functions g exactly, we replace them by a sample average over n independent days, obtained by simulation. Let ω represent the sequence of independent uniform random numbers that drives the simulation for those n days. When simulating the call center for different values of $\mathbf{y}$, we assume that the same uniform random numbers are used for the same purpose for all values of $\mathbf{y}$, for each day. That is, we use the same ω for all $\mathbf{y}$. Proper synchronization of these *common random numbers* is implemented by using a random number package with multiple streams and substreams (Law and Kelton 2000, L'Ecuyer et al. 2002, L'Ecuyer 2004).

The *empirical SL* over these n simulated days is a function of the staffing $\mathbf{y}$ and of ω. We denote it by $\hat{g}_{n,k,p}(\mathbf{y},\omega)$ for call type k in period p; $\hat{g}_{n,p}(\mathbf{y},\omega)$ aggregated over period p; $\hat{g}_{n,k}(\mathbf{y},\omega)$ aggregated for call type k; and $\hat{g}_n(\mathbf{y},\omega)$ aggregated overall. For a *fixed* ω, these are all deterministic functions of $\mathbf{y}$. Instead of solving directly (P0), we solve its *sample-average approximation* (SP0$_n$) obtained by replacing the functions g in (P0) by their sample counterparts $\hat{g}$ (here, $\hat{g}$ stands for any of the empirical SL functions, and similarly for g).

We know that $\hat{g}_{n,k,p}(\mathbf{y})$ converges to $g_{k,p}(\mathbf{y})$ with probability 1 for each (k,p) and each $\mathbf{y}$ when $n \to \infty$. In this sense, (SP0$_n$) converges to (P0) when $n \to \infty$. Suppose that we eliminate a priori all but a *finite* number of solutions for (P0). This can easily be achieved by eliminating all solutions for which the total number of agents is unreasonably large. Let $\mathscr{Y}^*$ be the set of optimal solutions of (P0) and suppose that no SL constraint is satisfied exactly for these solutions. Let $\mathscr{Y}_n^*$ be the set of optimal solutions of (SP0$_n$). Then, the following theorem implies that for n large enough, an optimal solution to the sample problem is also optimal for the original problem. It can be proved by a direct adaptation of the results of Vogel (1994) and Atlason et al. (2004); see also Ceşik and L'Ecuyer (2007).

Theorem 1 *With probability 1, there is an integer $N_0 < \infty$ such that for all $n \geq N_0$, $\mathscr{Y}_n^* = \mathscr{Y}^*$. Moreover, under the mild assumption that the service-level estimators satisfy a standard large-deviation principle (see Assumption 1 in Ceşik and L'Ecuyer (2007)), there are positive real numbers α and β such that for all n,*

$$P[\mathscr{Y}_n^* = \mathscr{Y}^*] \geq 1 - \alpha e^{-\beta n}.$$

We solve (SP0$_n$) by the cutting plane method described earlier, with the functions g replaced by their empirical counterparts. The major practical difficulty is to obtain the subgradients $\bar{\mathbf{q}}$. In fact, the functions $\hat{g}$ in the empirical problem (computed by simulation) are not necessarily concave

for finite n, even in the areas where the functions g of (P0) are concave. To obtain a (tentative) subgradient $\bar{\mathbf{q}}$ of a function $\hat{g}$ at $\bar{\mathbf{y}}$, we use forward finite differences as follows. For $j = 1, \ldots, IP$, we choose an integer $d_j \geq 0$, we compute the function $\hat{g}$ at $\bar{\mathbf{y}}$ and at $\bar{\mathbf{y}} + d_j \mathbf{e}_j$ for $j = 1, \ldots, IP$, where $\mathbf{e}_j$ is the jth unit vector, and we define $\bar{\mathbf{q}}$ as the IP-dimensional vector whose jth component is

$$\bar{q}_j = [\hat{g}(\bar{\mathbf{y}} + d_j \mathbf{e}_j) - \hat{g}(\bar{\mathbf{y}})]/d_j. \tag{3}$$

In our experiments, we used the same heuristic as in Ceşik and L'Ecuyer (2007) to select the d_j's: We took $d_j = 3$ when the SL corresponding to the considered cut was less than 0.5, $d_j = 2$ when it was between 0.5 and 0.65, and $d_j = 1$ when it was greater than 0.65. When we need a subgradient for a period-specific empirical SL ($\hat{g}_p$ or $\hat{g}_{k,p}$), the finite difference is formed only for those components of $\mathbf{y}$ corresponding to the given period; the other elements of $\bar{\mathbf{q}}$ are set to zero. This heuristic introduces inaccuracies, because $\hat{g}_p$ and $\hat{g}_{k,p}$ depend in general on the staffing of all periods up to p or even $p+1$, but it reduces the work significantly.

Computing $\bar{\mathbf{q}}$ via (3) requires $IP + 1$ simulations of n days each. This is by far the most time-consuming part of the algorithm. Even for medium-size problems, these simulations can easily require an excessive amount of time. For this reason, we use yet another important short-cut: We generally use a smaller value of n for estimating the subgradients than for checking feasibility. (The latter requires a single n-day simulation experiment.) That is, we compute each $\hat{g}(\bar{\mathbf{y}} + d_j \mathbf{e}_j)$ in (3) using $n_0 < n$ days of simulation, instead of n days. In most of our experiments (including those reported in this paper), we have used $n_0 \approx n/10$.

With all these approximations and the simulation noise, we recognize that the vector $\bar{\mathbf{q}}$ thus obtained is only a *heuristic guess* for a subgradient. It may fail to be a subgradient. In that case the cut (2) may remove feasible staffing solutions including the optimal one, and this may lead our algorithm to a suboptimal schedule; Atlason et al. (2004) and Ceşik and L'Ecuyer (2007) give examples of this. For this reason, it is a good idea to run the algorithm more than once with different streams of random numbers and/or slightly different parameters, and retain the best solution found.

At each step of the algorithm, after adding new linear cuts, we solve a relaxation of (SP0$_n$) in which the SL constraints have been replaced by a set of linear constraints. This is an integer programming (IP) problem.. But when the number of integer variables is large, we just solve it as a linear program (LP) instead, because solving the IP becomes too slow. To recover an integer solution, we select a threshold δ between 0 and 1; then we round up (to the next integer) the real numbers whose fractional part is larger than δ and we truncate (round down) the other ones. These two versions of the CP algorithm are denoted CP-IP and CP-LP.

When we add new cuts, we give priority to the cuts associated with the global SL constraints, followed by aggregate ones specific to a call type, followed by aggregate ones specific to a period, followed by the remaining ones. This is motivated by the intuitive observation that the more aggregation we have, the smoother is the empirical SL function, because it involves a larger number of calls. So its gradient is less likely to oscillate and the vector $\mathbf{q}$ defined earlier is more likely to be a

subgradient. Moreover, in the presence of abandonments, the SL functions tend to be non-concave in the areas where the SL is very small, and very small SL values tend to occur less often for the aggregated measures than for the more detailed ones that were averaged. Adding cuts that strengthen the aggregate SL often helps to increase the small SL values associated with specific periods and call types.

After adding enough linear cuts, we eventually end up with a feasible solution for ($SP0_n$). This solution may be infeasible for (P0) (because of random noise, especially if n is small) or may be feasible but suboptimal for (P0) (because one of the cuts may have removed the optimal solution of (P0) from the feasible set of ($SP0_n$)). To try improving our solution to ($SP0_n$), we perform a local search around it. In the CP-LP version, before launching this local search, the solution must be rounded to integers. This is done using a threshold δ as explained earlier. To determine this threshold, we perform a binary search over the interval $[0, 1]$, up to an accuracy of 0.01, to find the largest value of δ that yields a feasible integer solution for ($SP0_n$).

The local search proceeds by iteratively considering longer simulations to check the feasibility of the solutions that it examines. The number of days used in these simulations, n_1, starts from a value n_2 (smaller than n) specified as an input parameter and increases at each iteration by 50% of this value. Each iteration of the local search tries to solve $SP0_{n_1}$, in three phases... In the first phase, the current solution is checked again for feasibility with the new value of n_1 and agents are added at minimum cost until feasibility has been restored, if required. In the second phase, we attempt to reduce the cost of the solution by removing one shift at a time, until none of the possibilities is feasible. We further attempt to reduce the cost in the third phase by iteratively considering *switch moves* in which we try to replace an agent/shift pair by another one with smaller cost; the candidates for the switch moves are drawn at random, at each step, and the phase terminates when a maximum number of consecutive moves without improvement is reached. After the third phase, the current solution is tested for feasibility in a simulation of duration $n_3 = \max(n, 500)$ days. If it is feasible or if a time limit has been reached, the local search terminates, otherwise n_1 is increased and a new iteration is performed. Thus, at the end of the local search procedure, we have a feasible solution for either $SP0_{n_1}$ or $SP0_{n_3}$. The reason for using shorter, but increasingly long, simulations in the local search is the need to find some balance between limiting the time required to evaluate a large number of candidate solutions and ensuring the feasibility of the solutions considered (it is pointless to spend time examining a large number of solutions if they all turn up to be infeasible).

If we start the cutting plane algorithm with a full relaxation of ($SP0_n$) (no constraint at all), the optimal solution of this relaxation is $\mathbf{y} = \mathbf{0}$. The functions $\hat{g}$ are not concave at $\mathbf{0}$, and we cannot get subgradients at that point, so we cannot start the algorithm from there. As a heuristic to quickly remove this area where the staffing is too small and the SL is non-concave, we restrict the set of admissible solutions a priori by imposing (extra) initial constraints. To do that, we impose that for each period p, the skill supply of the available agents covers at least α_k times the total load for each

call type k (defined as the arrival rate of that call type divided by its service rate), where each α_k is a constant, usually close to 1. Finding the corresponding linear constraints is easily achieved by solving a max flow problem in a graph. See Cezik and L'Ecuyer (2007) for the details.

4 Computational Results

In order to assess the performance of the proposed algorithm, as well as the impact of flexibility on solutions, a number of problem instances were solved with the proposed algorithm and the two-step method. These instances were constructed so as to mimic the operations of real call centers. Their general setting is characterized as follows, unless stated otherwise.

The call center opens at 8:00 AM and closes at 5:00 PM; the working day is divided into $P = 36$ 15-minute periods. Shifts vary in length between 6.5 hours (26 periods) and 9 hours (36 periods) and include a 30-minute lunch break near the middle and two 15-minute coffee breaks (one pre-lunch and one post-lunch). Overall, there are 285 possible shifts (see Table 1).

Call arrivals are assumed to obey a stationary Poisson process over each period, for each call type, and independent across call types. The profile of the arrival rates in the different periods are inspired from observations in real-life call centers at Bell Canada (Avramidis et al. 2004). They are plotted separately for each instance. All service times are exponential with service rate $\mu = 8$ calls per hour. Patience times have a mixture distribution: the patience is 0 with probability 0.001, and with probability 0.999, it is exponential with rate 0.1 per minute. The *routing policy* is an agents' preference-based router (Buist and L'Ecuyer 2005).

For most instances, we only consider *aggregate* service level constraints for each period. These require that at least 80% of all the calls received during the period be answered within 20 seconds (i.e., we have $\tau_p = 20$ seconds and $l_p = 0.8$ for each p). The satisfaction of these constraints implies that the global constraint with $\tau = 20$ seconds and $l = 0.8$ is automatically satisfied, but we still require this explicitely, because this constraint plays a key role in the cutting-plane algorithm. In some cases, we also impose *disaggregate* SL constraints for each (call type, period) combination (k,p) with $\tau_{k,p} = 20$ seconds and $l_{k,p} = 0.5$ for all k and p. Note that this in turn implies the satisfaction of aggregate SL constraints for each call type k with $\tau_k = 20$ seconds and $l_k = 0.5$.

The formula used to compute agents' costs accounts for both the number of skills in the agent's skill set and the length of the shift being worked:

$$c_{iq} = (1 + (\eta_i - 1)\varsigma) l_q / 30 \quad \text{for all } i \text{ and } q, \tag{4}$$

where l_q is the length (in periods) of shift q, 30 is the number of periods in a "standard" 7.5-hour shift, η_i is the cardinality of S_i, and ς is an instance-specific parameter that represents the cost associated with each agent skill.

We first compare the two solution methods described (i.e., TS and CP) on three instances that correspond to a small (section 4.1), a medium-sized (section 4.2) and a larger call center (section

Type	length	shift start	break1 start	lunch start	break3 start
1	7:30	8:00, 8:30, 9:00, 9:30	9:30-11:30	12:00, 12:30, 13:00	14:00-15:30
2	7:45	9:15	10:45-11:15	12:00, 12:30, 13:00	14:00-15:30
3	8:00	9:00	10:30-11:00	12:00, 12:30, 13:00	14:00-15:30
4	8:15	8:45	10:15-10:45	12:00, 12:30, 13:00	14:00-15:30
5	8:30	8:30	10:00-10:30	12:00, 12:30, 13:00	14:30-16:00
6	9:00	8:00	11:00-12:00	13:00, 13:30, 14:00	14:45-16:15
7	6:30	10:00	11:30-12:00	13:00, 13:30, 14:00	14:15-15:45

Table 1: Description of the 285 shifts for our examples

4.3). For the medium-size center, two variants are considered: M_1, in which only aggregate SL constraints considered, and M_2 with aggregate and disaggregate SL constraints. For the larger center, we also examine the impact of having a longer working day.

Both TS and CP use Cplex 9.0 to solve the optimization problems. To allow a fair comparison of the methods, we allocate the same CPU time "budget" to each. Considering the nature of the algorithms, this cannot be done by simply stopping them when this time limit is reached. Instead, we must carefully adjust, by trial and error, the number of simulated days n, which is a key parameter of both methods, to obtain running times close to the target budget. It is clear that one would not use such a procedure in a practical context, but this is necessary for the comparative study. For each instance, we consider several different budgets, since we expect that a higher value of n will produce more accurate and more stable results. Furthermore, in each case, r replications of each method/budget combination are performed to account for the random elements in both methods.

In the first phase of TS, to simulate each individual period of the call center and evaluate the results of the simulations, we use the batch means method (Law and Kelton 2000). Each batch is constituted by a minimum number of 30 simulation time units and statistical observations are collected on a minimum of 50 batches, using 2 warmup batches before starting to collect statistics.

Final solutions obtained by the two methods were simulated for $n_* = 50,000$ days as an additional (much more stringent) feasibility test, and each solution was declared feasible or not according to the result of this test, i.e., according to the feasibility of (SP0$_{n_*}$).

For each instance (or variant), results are summarized in a table with the following column headings: *case*, an index assigned for a specific CPU time budget; n, the number of simulated days for checking feasibility when adding cutting planes and for the local search at the end of the algorithm; n_2, the starting number of days in local search simulations; CPU_{avg}, the average CPU time per replication; *Min cost* and *Med cost*, which are respectively the minimum and median costs of all solutions (feasible or not) obtained by this method over the r replications; P^*, the percentage of replications that returned a feasible solution for (SP0$_{n_*}$); and P_1^*, the percentage that returned a feasible solution with cost within 1% of the best known feasible solution (the lowest-cost feasible solution for (SP0$_{n_*}$) generated by either algorithm, over all replications and CPU time budgets, in all experiments that

we have done, including those described in Section 4.4). We also report the maximum relative violation gap (in percent) observed in a SL constraint for each type of constraints; G_{period} and $G_{\text{call,period}}$ refer respectively to violations of SL constraints for periods and for individual (call type, period) combinations.

4.1 A small call center

This instance has $K = 2$ call types and $I = 2$ agent types, with $S_1 = \{1\}$ and $S_2 = \{1,2\}$. Agent costs are computed by setting the parameter ς equals to 0.2 in formula 4. Arrival rates for the two call types are plotted in Figure 1. All SL constraints are enforced in this instance. Four different CPU time budgets were considered: 3, 15, 30 and 60 minutes... Results, based on $r = 32$ replications, are displayed in Table 2.

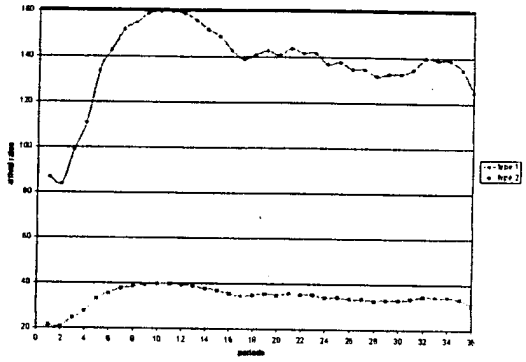


Figure 1: Small center: arrival rates

Case	Algorithm	n	n_2	CPU_{avg} sec.	Min cost	Med cost	P_1^*	P^*	G_{period}	$G_{\text{call,period}}$
1	CP-IP	120	50	136	36.31	36.91	0	100	0	0
	TS	120		162	35.60	35.60	0	0	0.81	0
2	CP-IP	1500	800	914	35.13	35.17	100	100	0	0
	TS	1500		898	35.59	35.59	0	0	0.94	0
3	CP-IP	1600	1000	1774	35.03	35.17	75	87	0	0.15
	TS	2800		1753	35.67	35.67	0	0	0.79	0
4	CP-IP	2000	1000	3694	35.03	35.17	100	100	0	0
	TS	6000		3453	35.67	35.67	0	0	0.79	0

Table 2: Small center: results obtained with CP-IP and TS for different CPU time budgets

Several observations can be made from Table 2. First, CP-IP is almost always able to find feasible solutions to the problem and most of them are very good. The only case where solutions are more expensive than those obtained with TS, and these are infeasible, is when $n = 120$, a rather small

value. On the other hand, the results for the three larger values of n are quite similar and setting n to 1500 is probably sufficient for this small center. It is also interesting, and surprising, to find out that all runs of TS failed to find a feasible solution, even though constraint violations were always inferior to 1%. The solutions produced by TS are also more expensive than the ones obtained with CP for large enough n. Such an outcome in not surprising considering the inherent shortcomings of the method. A closer examination of the distribution of the cost of solutions reveals that, except for CP-IP with $n = 120$, these costs vary very little. With TS, all runs with the same computing budget return in fact the same cost value. In practice, a manager might be willing to use almost-feasible solutions, considering the fact that the center will always experience stochastic variation in the arrival process and the SL in any case. For this reason, it is probably useful to report slightly infeasible solutions in general, and not only the feasible ones.

The best scheduling solutions obtained by CP and TS are described in Table 3. In this table, shift types correspond to the length of the shifts as indicated in Table 1. Both methods return solutions with the same number of agents (31), most of which are specialists (type 1). Further analysis of these solutions reveals that they differ only slightly in terms of the duration of the shifts scheduled, but CP uses more specialists than TS (24 vs 21) and thus gives a cheaper solution.

Algorithm	Agent type	Shift type						
		1	2	3	4	5	6	7
CP-IP	1	6	3	3	1	1	8	2
	2	3	0	0	1	1	2	0
TS	1	6	1	3	1	2	7	1
	2	4	1	0	1	0	3	1

Table 3: Small center: scheduling solutions

Service levels for the best solution obtained are plotted in Figure 2. We see (1) a wide variation of the SL throughout the day and (2) that calls of type 1 have much better SL than those of type 2. This imbalance can be explained by the fact that the type 1 calls can be answered by less expensive specialists, while type 2 calls must be handled by generalists. This observation highlights the fact that to ensure a fair treatment of all call types in a real-life setting, it is often necessary to include call-type specific SL constraints (either over the whole day or for each period) in the problem formulation.

4.2 A medium-sized call center

In the medium-sized instances, there are $K = 5$ call types and $I = 15$ agent types. Five of the agents types are specialists handling a single call type, while the ten other types are generalists handling between 2 and 5 call types. Details on skill sets can be found in Avramidis et al. (2007b). The parameter ς used to compute agent costs in formula 4 is now equal to 0.1. Arrival rates for all call types are plotted in Figure 3.

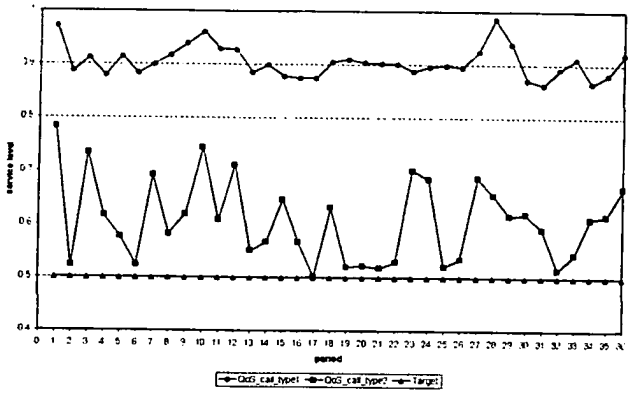


Figure 2: Small center: service levels by period

As mentioned earlier, we consider two variants of this example: in M_1, only global and per-period SL constraints are enforced, while M_2 also includes disaggregate SL constraints. Since in practice one may find it hard to satisfy all SL constraints and since real-life call center managers are often more interested by global SL, it seemed interesting to compare these two variants. For each of them, we performed $r = 8$ replications for several CPU time budgets of 15, 30 and 60 minutes. Running CP-IP would have taken unacceptable running times for these large problem instances, so we used CP-LP instead. The results for M_1 and M_2 are summarized in Tables 4 and 5. We find that most replications of CP-LP return low-cost feasible solutions for both variants, but the cost variation between solutions is more pronounced than for the small center. The quality of solutions also increases significantly with n, which emphasizes the importance of performing long enough simulations to obtain good results. Contrary to what was observed for the small center, TS always finds feasible solutions, but their cost is much higher than the cost of CP solutions; in fact, the optimality gap with respect to the best solution found is over 25% for M_1 and close to that value for M_2. This shows that large suboptimality gaps with TS do occur in realistic call center settings, and not only in artificial examples.

An examination of the best scheduling solutions obtained for M_1 with CP and TS reveals that CP not only uses significantly fewer agents (14 vs 17), but also that these agents have on average a smaller number of skills (2..50 vs 2.76), and work shorter hours (more than 10 minutes less per day, on average). Similar results are also observed for M_2. The best solution found by CP for M_1 is cheaper than the one found for M_2; this was expected since M_1 is a relaxation of M_2. In fact, the most interesting conclusion that one could draw here is that the increase in cost incurred when imposing the disaggregate SL constraints is rather marginal.

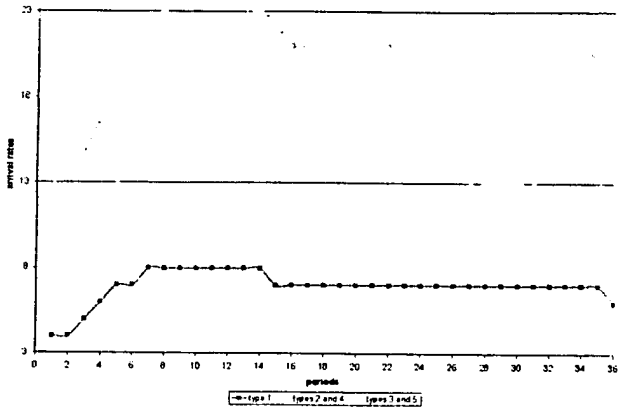


Figure 3: Medium-sized center: arrival rates

Case	Algorithm	n	n_2	CPU_{avg} sec.	Min cost	Med cost	P_1^*	P^*	G_{period}
1	CP-LP	300	100	855	18.33	18.85	0	87	0.51
	TS	1200		897	21.83	21.83	0	100	0
2	CP-LP	600	100	1598	17.56	18.43	0	75	0.41
	TS	2400		1774	21.72	21.72	0	100	0
3	CP-LP	1000	400	2686	17.36	17.66	25	87	0.02
	TS	3000		2793	21.69	21.69	0	100	0

Table 4: M_1: results obtained with CP-LP and TS for different CPU time budgets

4.3 A larger call center

The larger center instances have $K = 20$ call types and $I = 35$ agent types. Twenty of the agents types are specialists handling a single call type, while the fifteen other types are generalists handling between 4 and 9 call types. Details on skill sets can be found in Avramidis et al. (2007b). Agent costs are computed with $\varsigma = 0.1$. We only consider SL constraints by period, plus the global SL constraint. Two CPU time budgets are examined: 5 and 10 hours. We use the CP-LP version of CP and perform $r = 8$ replications.

One of our objectives with this example is to show that the performance of CP does not depend much on the particular structure of the shifts. We thus consider again two variants: L_{36}, which uses the 9-hour working day and the same shift structure as the previous examples, and L_{52}, which has a working day starting at 8:00 AM and ending at 9:00 PM; in that variant, the total number of periods is 52 and all the shifts have a fixed length of 7.5 hours, thus yielding a total of 123 different shifts (considering also shifts starting at 1:00 PM and 1:30 PM in order to cover the additional periods). Arrival rates for the L_{36} variant and the 36 first periods of L_{52}, follow exactly the same pattern as in

Case	Algorithm	n	n_2	CPU_{avg} sec.	Min cost	Med cost	P_1^*	P^*	G_{period}	$G_{call.period}$
1	CP-LP	400	100	828	18.48	18.98	0	87	0.58	0
	TS	600		876	21.58	21.58	0	100	0	0
2	CP-LP	400	100	1751	18.11	18.97	0	87	0.54	0
	TS	1500		1677	21.65	21.65	0	100	0	0
3	CP-LP	400	100	3520	17.92	18.57	0	75	0.28	0
	TS	3000		3212	21.80	21.80	0	100	0	0

Table 5: M_2: results obtained with CP and TS varying the CPU time budget

the medium-size example (see Figure 3), with different scalings for the different call types; for L_{52} they then decrease slowly during the last 16 periods. The results for L_{36} are displayed in Table 6.

Here, CP has difficulty finding a feasible solution with the smallest computing budget: only two of the eight runs returned feasible solutions, even though constraint violations might not be severe. This situation is clearly alleviated by allotting more CPU time. With more time, CP also finds significantly better solutions. TS always finds feasible solutions, but the solutions returned are on average 20% more expensive than those obtained with CP. This confirms our observations of the previous section regarding the poor performance of TS. Surprisingly, increasing the CPU budget does not improve the performance of TS. In fact, with more time, it returns inferior solutions. This is because the method obtains a different staffing solution in the first step; while this solution might track more closely the call arrival curve, it ends up leading to a poorer scheduling solution. A close examination of the relative cost distributions of solutions highlights the fact that, in the case of TS, all replications produce identical, and largely suboptimal, solutions. On closer examination of the best scheduling solutions obtained by the two methods, we find that the CP solution is less expensive because it covers the demand with only 52 agents compared to 62 for TS.

Case	Algorithm	n	n_2	CPU_{avg} min.	Min cost	Med cost	P_1^*	P^*	G_{period}
1	CP-LP	400	50	288	82.02	82.20	0	25	0.25
	TS	1500		299	96.08	96.08	0	100	0
2	CP-LP	500	50	577	78.87	81.80	25	87	0.15
	TS	2400		598	102.87	102.87	0	100	0

Table 6: L_{36}: results obtained with CP-LP and TS for different CPU time budgets

The results for the L_{52} variant are reported in Table 7. On this larger problem, just one of the 8 solutions found by CP-LP was declared feasible by the 50,000-day simulation, but it is an excellent solution with a cost of 131.7. With more time, CP returned 4 feasible solutions out of 8 runs, but these turned out to be inferior to the one found in 5 hours, probably due to simulation noise. Overall, these results emphasizes the importance of performing several trial runs when using this

type of approach.

Case	Algorithm	n	n_2	CPU_{avg} minutes	Min cost	Med cost	P_1^+	P^+	G_{period}
1	CP-LP	300	50	296	130.8	133.6	12	12	0.54
	TS	1500		262	156.1	156.1	0	100	0
2	CP-LP	400	50	588	133.5	137.7	0	50	0.44
	TS	1800		542	156.1	156.1	0	100	0

Table 7: L_{52}: results obtained with CP-LP and TS for different CPU time budgets

All the solutions returned by TS were declared feasible, but they are significantly more expensive, but they are again highly suboptimal, with a cost of 156.1. When we examine the best solutions found by CP-LP and by TS for the L_{52} instance, we first remark that CP-LP uses only 96 agents compared to 107 for TS. We also note that 15 agents in the CP solution are specialists, while there are none in the TS solution. Furthermore, TS uses 32 expensive generalists with 7 skills compared to 12 for CP. These three factors combined explain the large difference in cost.

Our motivation for investigating the 52-period example was to verify that CP performed correctly for instances with a different shift structure. Our results confirm this, but at the same time they highlight one of the potential shortcomings of the approach, which is that, because of simulation noise, when there is a large number of constraints, one often ends up with no feasible solution, even though several near-feasible solutions may have been identified. We address this issue in the next subsection.

4.4 Getting more feasible results

Empirical results show that, as problem instances become larger and more complex, there is a definite possibility that CP would return a set of low-cost, but only nearly-feasible solutions. While this may be acceptable in some practical settings, it is nonetheless annoying to be unable to provide the call center manager with a solution that meets all his/her requirements. A simple and attractive way of tackling this problem consists in slightly increasing the right-hand side value of the SL constraints when applying the algorithm (except obviously for the final long simulation that is used to determine the feasibility of solutions). It should be noted that this idea is not specific to the CP procedure and could therefore be applied with any other solution approach.

We first tested this idea on the L_{52} instance, using values of 0.81 and 0.82 as target SL for all periods. We combined these tests with experiments on the value of the threshold δ used for rounding continuous solutions to integer ones in CP-LP. The rationale for investigating different values of δ is that the rounding procedure introduces a heuristic element in what would otherwise be an exact procedure and that selecting the best value for this threshold is far from obvious.

In our experiments, we considered three different values of δ (0.5, 0.6 and 0.7) for CPU budgets of 5 and 10 hours, and we ran 8 replications in each case. All 48 runs performed for each of the two target values turned out to be feasible for the 50,000-day simulation! Furthermore, one of these runs returned a better solution than the best one we had before (cost of 130.5 vs 131.7). The results obtained for a SL target of 0.81 are summarized in Table 8. These results show that the value selected for δ seems to have a slight, but not critical impact on the quality of the solutions obtained. In fact, it seems to be much more important to make sure that the runs that are made do produce feasible solutions, in order to have a larger set to choose from. We then ran CP-LP with the original

δ	CPU budget hours	Min cost	Med cost	Worst SL
0.5	5	132.10	134.85	≥ 0.8
	10	132.30	136.80	≥ 0.8
0.6	5	130.50	133.55	≥ 0.8
	10	135.30	138.60	≥ 0.8
0.7	5	132.50	135.20	≥ 0.8
	10	131.60	138.05	≥ 0.8

Table 8: L_{52}: results obtained with a target SL of 0.81

0.80 target value for different values of δ. These tests clearly showed that modifying δ alone was not sufficient to consistently obtain feasible solutions, since more than half of these runs returned infeasible solutions. However, they did allow us to find an even cheaper feasible solution with a cost 130.4. We also ran the algorithm with a target SL value of 0.81 for the other instances (keeping the value of δ unchanged at 0.5). The results can be summarized as follows:

- For the small center, all runs returned feasible solutions, which is not surprising considering previous results, but these solutions were no better than the ones obtained previously.
- For the medium-sized instances, all runs produced feasible solutions, and improved solutions were obtained both for the M_1 and M_2 instances (with respective costs of 17.22 and 17.39).
- For the L_{36} instance, 15 of the 16 runs yielded a feasible solution, but never better than the best one found with an SL target of 0.8.

Overall, slightly increasing the value of the SL target value is a useful device for making sure that the method will return feasible solutions. However, there is no guarantee that better solutions will be found by doing so. The variability of our results highlights once again the stochastic nature of the algorithm, which cannot be avoided considering the significant amount of noise in the simulations.

4.5 The impact of flexibility

We performed another series of numerical experiments to quantify empirically the impact of the flexibility provided by a rich set of shift types. Those experiments were performed on the small center of subsection 4.1. We considered three sets of shift types: the original one with all 285 shifts, a slightly reduced one with 267 shifts, obtained by deleting the 26-period and some 36-period shifts, and adding some 35-period ones, and finally a much more restricted set in which we only allow the 105 7.5-hour shifts. The staffing solutions corresponding to the best scheduling solutions obtained for these three cases are plotted in Figure 4, along with the optimal staffing solution computed by considering each period individually. Three main conclusions can be drawn from this figure:

1. As shown by the solution with 285 shifts, if enough flexibility is introduced in the set of available shifts, it is possible to find schedules that track closely the staffing requirements.

2. Even a slight decrease in flexibility (e.g., by going from 285 to 267 shifts) can lead to a significant overstaffing in some periods.

3. Schedules with a relatively small number of fixed-length shifts (the 105-shift case) are bound to suffer from major overstaffing.

It follows that, while the complexity of the scheduling problem significantly increases with the number of available shifts, there are definite benefits to be reaped from the introduction of more varied shift types.

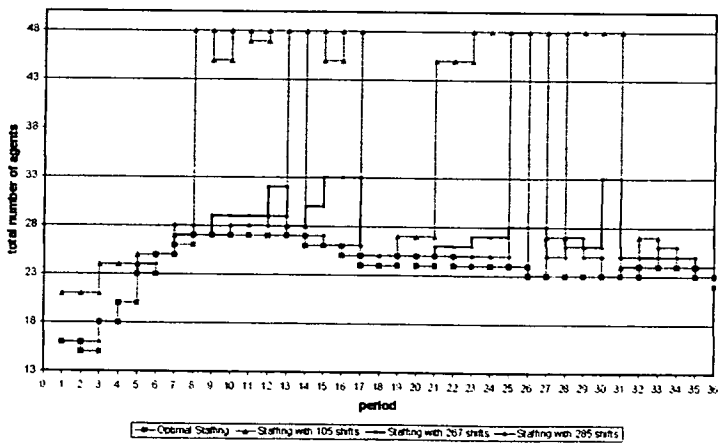


Figure 4: Small center: staffing solutions with 105, 267 and 285 shift types

5 Conclusion

We have proposed in this paper a simulation-based methodology to optimize agent scheduling over one day in a multiskill call center. Even though the use of common random numbers reduces the simulation noise (or variance) significantly, there is still a fair amount of randomness in the solution provided by the algorithm, mainly due to the fact that the simulation length must be kept short (because the estimation of each subgradient requires simulations at up to thousands of different parameter values). Yet, to our knowledge, better solutions are found with this approach than with any other method we know. In particular, during the development of the cutting plane algorithm, we also implemented simultaneously a metaheuristic method based on neighborhood search combined with queueing approximation, along the lines of Avramidis et al. (2006), but we were unable to make it competitive for solving the scheduling problem.

In practice, one may run the algorithm a few times (e.g., overnight) to obtain a few solutions and retain the best found. We also showed that by slightly perturbing the SL targets, it is possible to overcome some of the problems caused by the presence of the simulation noise and thus to greatly increase the probability of obtaining feasible, high-quality solutions.

Future research on this problem include the search for faster ways of estimating the subgradients (by simulation), refining the algorithm to further reduce the noise in the returned solution, and extending the technique to simultaneously optimize the scheduling and the routing of calls (via dynamic rules).

Acknowledgements

This research has been supported by Grants OGP-0110050, OGP38816-05, and CRDPJ-320308 from NSERC-Canada, and a grant from Bell Canada via the Bell University Laboratories, to the second and third authors, and a Canada Research Chair to the third author. The fourth author benefited from the support of the University of Calabria and the Department of Electronics, Informatics and Systems (DEIS), and her thanks also go to professors Pasquale Legato and Roberto Musmanno.

References

J. Atlason, M. A. Epelman, and S. G. Henderson. Call center staffing with simulation and cutting plane methods. *Annals of Operations Research*, 127:333–358, 2004.

A. N. Avramidis, W. Chan, and P. L'Ecuyer. Staffing multi-skill call centers via search methods and a performance approximation. Submitted, Revised in May 2007, 2006.

A. N. Avramidis, A. Deslauriers, and P. L'Ecuyer. Modeling daily arrivals to a telephone call center. *Management Science*, 50(7):896–908, 2004.

A. N. Avramidis, M. Gendreau, P. L'Ecuyer, and O. Pisacane. Simulation-based optimization of agent scheduling in multiskill call centers. In *Proceedings of the 2007 Industrial Simulation Conference*. Eurosis, 2007a.

A. N. Avramidis, M. Gendreau, P. L'Ecuyer, and O. Pisacane. Simulation-based optimization of agent scheduling in multiskill call centers. Technical report, Technical Report CIRRELT-2007-44, CIRRELT, University of Montreal, 2007b. Preprint.

S. Bhulai, G. Koole, and A. Pot. Simple methods for shift scheduling in multi-skill call centers, 2007. Manuscript, available at `http://www.cs.vu.nl/~koole/research`.

L. Brown, N. Gans, A. Mandelbaum, A. Sakov, H. Shen, S. Zeltyn, and L. Zhao. Statistical analysis of a telephone call center: A queueing-science perspective. *Journal of the American Statistical Association*, 100:36–50, 2005.

E. Buist and P. L'Ecuyer. A Java library for simulating contact centers. In *Proceedings of the 2005 Winter Simulation Conference*, pages 556–565. IEEE Press, 2005.

U. D. o. L. Bureau of Labor Statistics. *Occupational Outlook Handbook, Customer Service Representatives, 2006-07 Edition*. 2007. Available online at `http://www.bls.gov/oco/ocos280.htm` (last accessed February 14, 2007).

M. T. Cezik and P. L'Ecuyer. Staffing multiskill call centers via linear programming and simulation. *Management Science*, 53, 2007. To appear.

CRTC. Final standards for quality of service indicators for use in telephone company regulation and other related matters, 2000. Canadian Radio-Television and Telecommunications Commission, Decision CRTC 2000-24. See `http://www.crtc.gc.ca/archive/ENG/Decisions/2000/DT2000-24.htm`.

N. Gans, G. Koole, and A. Mandelbaum. Telephone call centers: Tutorial, review, and research prospects. *Manufacturing and Service Operations Management*, 5:79–141, 2003.

L. V. Green, P. J. Colesar, and J. Soares. An improved heuristic for staffing telephone call centers with limited operating hours. *Production and Operations Management*, 12:46–61, 2003.

S. Halfin and W. Whitt. Heavy-traffic limits for queues with many exponential servers. *Operations Research*, 29:567–588, 1981.

S. Henderson and A. Mason. Rostering by iterating integer programming and simulation. In *Proceedings of the 1998 Winter Simulation Conference*, volume 1, pages 677–683, 1998.

A. Ingolfsson, E. Cabral, and X. Wu. Combining integer programming and the randomization method to schedule employees. Technical report, School of Business, University of Alberta, Edmonton, Alberta, Canada, 2003. Preprint.

O. B. Jennings, A. Mandelbaum, W. A. Massey, and W. Whitt. Server staffing to meet time-varying demand. *Management Science*, 42(10):1383–1394, 1996.

E. G. Keith. Operator scheduling. *AIIE Transactions*, 11(1):37–41, 1979.

G. Koole and A. Mandelbaum. Queueing models of call centers: An introduction. *Annals of Operations Research*, 113:41–59, 2002.

A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, New York, third edition, 2000.

P. L'Ecuyer. *SSJ: A Java Library for Stochastic Simulation*, 2004. Software user's guide, Available at `http://www.iro.umontreal.ca/~lecuyer`.

P. L'Ecuyer, R. Simard, E. J. Chen, and W. D. Kelton. An object-oriented random-number package with many long streams and substreams. *Operations Research*, 50(6):1073–1075, 2002.

V. Mehrotra. Ringing up big business. *ORMS Today*, 24(4):18–24, October 1997.

A. Pot, S. Bhulai, and G. Koole. A simple staffing method for multi-skill call centers, 2007. Manuscript, available at `http://www.cs.vu.nl/~koole/research`.

R. Stolletz and S. Helber. Performance analysis of an inbound call center with skill-based routing. *OR Spectrum*, 26:331–352, 2004.

G. M. Thompson. Labor staffing and scheduling models for controlling service levels. *Naval Research Logistics*, 8:719–740, 1997.

S. Vogel. A stochastic approach to stability in stochastic programming. *Journal of Computational and Applied Mathematics*, 56:65–96, 1994.